Development of an Autonomous Deorbiting Device for a CubeSat

by

Ashley Kendal Naudé

Thesis presented in partial fulfilment of the requirements for the degree of Master in Engineering in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof. WH Steyn

March 2020

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2020

Abstract

The CubeSat industry is a budding one in the space sector. CubeSats are quickly becoming the satellite of choice for many Earth Observation missions and even interplanetary missions. With the CubeSats chiefly launched to Low Earth Orbit, the problem of overcrowding this space is looming. Low Earth Orbit is home to many larger satellites and is littered with space debris. If these CubeSats continue to be launched at the rate that they are currently being launched at, there will be a severe problem in a few years when they are non-functional and stuck in orbit. Collisions between space objects creates thousands of smaller pieces of debris which exponentially increases the probability of more collisions. It is, therefore, necessary to implement a plan to prevent more debris from forming in Low Earth Orbit, to prevent the runaway scenario of debris formations as well as to keep the space available for new satellites to be launched in future.

This thesis focuses on the design and implementation of a CubeSat deorbiting device to be placed in the CubeSat for use at the end of its mission lifetime. First, a literature review and any theoretical knowledge needed for this project are given. A low-power hardware prototype is then designed and implemented in both vacuum chamber and atmospheric level tests. Simulations are done in MATLAB and Simulink to determine the size of the deorbit device needed for different CubeSat sizes and orbit heights. The simulations also include different device designs and satellite attitude states. Finally, the hardware tests' results are discussed and the different simulations compared.

Uittreksel

Die CubeSat-industrie is 'n ontluikende industrie in die ruimtesektor. CubeSats word vinnig die gekose satelliet vir baie Aardwaarnemings-missies en selfs interplanetêre missies. Met die CubeSats wat hoofsaaklik na Lae Aard-wentelbane gelanseer is, is die probleem om die ruimte te oorbevolk 'n bedreiging. Lae Aard-wentelbane is die tuiste van baie groter satelliete en baie ruimterommel. As hierdie CubeSats steeds gelanseer word teen die tempo waarteen hulle tans gelanseer word, sal daar oor 'n paar jaar 'n ernstige probleem wees wanneer hulle nie funksioneel is nie en in die ruimte beset. Botsings tussen ruimte-voorwerpe skep duisende kleiner stukke rommel wat eksponensieel die waarskynlikheid van meer botsings verhoog. Dit is dus nodig om 'n plan in werking te stel om te voorkom dat meer rommel in die Lae Aarde-wentelbaan vorm, om die wegholscenario van rommelformasies te voorkom, asook om die ruimte beskikbaar te stel vir nuwe satelliete wat in die toekoms gelanseer kan word.

Hierdie tesis fokus op die ontwerp en implementering van 'n CubeSat-ontwentelapparaat wat in die CubeSat geplaas moet word vir gebruik aan die einde van sy lewensduur. Eerstens word 'n literatuuroorsig en alle teoretiese kennis wat vir hierdie projek benodig word, gegee. Daarna word 'n lae-krag hardeware prototipe ontwerp en geïmplementeer in vakuumtenk en atmosferiese toetse. Simulasies word in MATLAB en Simulink gedoen om die grootte van die ontwentelsapparaat wat benodig word vir verskillende CubeSat-groottes en wentelbaanhoogtes te bepaal. Die simulasies bevat ook verskillende apparaatontwerpe en toestande vir satellietoriëntasie. Laastens word die resultate van die hardewaretoetse bespreek en die verskillende simulasies vergelyk.

Acknowledgements

I would like to express my appreciation and gratitude to the following people for their support throughout the project:

- My supervisor, Prof. WH Steyn, for his guidance, patience and willingness to share his immense knowledge with me.
- SANSA, the South African National Space Agency, for providing me with financial aid in order to do my Master's.
- Mr Wynand van Eeden for assisting with the PCB cutting and for always being a helping hand for small hurdles I may have encountered.
- Mrs Jenny Martin for making the process of ordering parts and components as smooth as possible.
- My fellow ESL colleagues, especially the "Backstreet Boys" and the "MICjaggers" for providing constant banter and support over these two years.
- My parents for never allowing me to give up.
- Bernard and Francisca, two of my best friends, for constantly providing emotional support and motivation, and checking up on me during the typing stages.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations and Acronyms

ADC	Analogue-to-Digital Converter
ADR	Active Debris Removal
AU	Astronomical Unit
COESA	US Committee on Extension to the Standard Atmosphere
CoM	Centre of Mass
CSD	Canisterized Satellite Dispenser
DCM	Direction Cosine Matrix
DIP	Dual In-line Package
ECI	Earth-Centred Inertial coordinates
EDT	Electro Dynamic Tether
EoL	End-of-Life
ESA	European Space Agency
GEO	Geosynchronous Orbit
init	Initialisation
ISS	International Space Station
JPL	Jet Propulsion Lab
LDEF	Long-Duration Exposure Facility
LEO	Low Earth Orbit
NASA	National Aeronautics and Space Administration
NRL	U.S. Naval Research Laboratory
ORC	Orbit Reference Coordinates
P-POD	Poly Picosatellite Orbital Deployer
PDR	Passive Debris Removal
PCB	Printed Circuit Board
RAAN	Right Ascension of the Ascending Node
rad	Radians
RPY	Roll, Pitch and Yaw
SBC	Spacecraft Body Coordinates
SOIC	Small-Outline Integrated Circuit
SRP	Solar Radiation Pressure
STELA	Semi-analytic Tool for End-of-Life Analysis software
TLE	Two-Line Element

UN	United Nations
US	United States (of America)
USART	Universal Synchronous Asynchronous Receiver-Transmitter
VSSOP	Very-thin Shrink Small-Outline Package

Symbols, Variables and Constants

Greek Letters

α	Incidence Angle of Solar Wind
β	Sun Angle to Satellite Normal
ε	Mechanical Energy
θ	Pitch Angle
μ	Body Gravitational Constant
ν	True Anomaly
π	Pi Constant
ρ	Atmospheric Density
φ	Roll Angle
ψ	Yaw Angle
Ω	RAAN
ω	Argument of Perigee; Angular Body Rates

Lower-case Letters

a	Semi-Major Axis; Acceleration; Spheroid x Radius
b	Spheroid y Radius
c	Centre; Speed of Light in Vacuum; Spheroid z Radius
e	Eccentricity
g	Gravitational Acceleration
h	Angular Momentum; Height
i	Orbital Inclination
l	Length
m	Mass
n	Normal
p	Semilatus Rectum; Solar Radiation Pressure Magnitude
q	Quaternion
r	Radius
t	Tangential
v	Velocity

Upper-case Letters

A	Area
E	Energy
F	Force
G	Newton's Universal Gravitational Constant
R	Equatorial Radius
S	Sun

Subscripts

a	Apogee
b	Satellite Body
D	Drag
E	Earth
G	Gravity
i	Inertial reference; Incidence
n	Normal
o	Orbit reference
OB	Oblateness
p	Perigee
r	Reflected
SRP	Solar Radiation Pressure
t	Tangential

Chapter 1

Introduction

The need for smaller, cheaper satellites brought about the CubeSat design in 1999. Today, with over one thousand launched CubeSats, they are quickly becoming the norm for satellite missions to Low Earth Orbits and occasionally beyond. With so many new satellites placed in space, the increased risk of collisions between orbiting objects, including active satellites as well as defunct satellites and space debris has also grown considerably. Space debris has increased over the years since the first satellite launched in 1957 and plans to decrease the amount of debris in space have only recently been adopted. Most of the debris is too small to track, which means collisions cannot be predicted and can destroy functional, active satellites, resulting in significant financial loss and wastage of resources. Orbital debris also poses a risk to the safety of astronauts while on space-walks, potentially piercing their suits. With every new collision comes new pieces of debris, creating a runaway snowball effect of more and more debris creation. It is, therefore, necessary to deorbit all these new small satellites entering space either purposefully at the end of mission life or autonomously when the satellite is no longer communicating or functional, to prevent new debris stuck in orbit.

1.1 Project Objectives

The first objective of this project is to design, make and test a small, low mass and volume device to deploy at End-of-Life (EoL) for a CubeSat to aid in the deorbiting thereof. The prototype device must be tested in vacuum conditions to ensure it can be operated in space. The device must be able to deorbit the satellites within 25 years from an initial altitude of at least 700 km. If the satellites are able to deorbit within 25 years unaided by a device, the device must accelerate the deorbit rate by at least a factor of ten. The prototype must be able to work regardless of the CubeSat's attitude and have a independent power source for deployment. The second and final objective of this project is to undergo simulation studies to support the choice of the prototype. The best device will be selected for the specific CubeSat size and orbit height based on the simulation results.

1.2 Thesis Outline

This subsection gives a short overview of the sections covered in the report:

Chapter 1: Introduction

This section introduces the purpose of the project, explaining briefly why it is necessary to deorbit satellites at EoL. This project's objectives are also briefly stated.

Chapter 2: Literature Study

This section presents the literature study for the project. It expands on the purpose given in the Introduction paragraph by starting with a brief history of CubeSats and their specifications. The problem of orbital debris is also studied, especially regarding CubeSats and the role they play in this contemporary issue.

Chapter 3: Theoretical Background

This section provides all the necessary background knowledge for this project. First, orbits and orbital dynamics will be discussed and thereafter the causes of satellite orbit decay. Atmospheric drag effects will be the initial cause presented, along with the different atmospheric density models available. Subsequently, gravity and the Earth's oblateness will be looked into as further causes of orbit decay. Finally, solar radiation pressure will be studied.

Chapter 4: Hardware Description

This section covers the design process followed for the project. It discusses the choices made during the design process and explains why the final prototype was chosen. It also covers the methods of testing the hardware in atmospheric conditions as well as in a vacuum set-up.

Chapter 5: Simulation Environments

This section introduces the MATLAB and Simulink simulation used to compare three CubeSats of different sizes at three different orbit heights to determine the best deorbiting device per situation. The satellites at a specific orbit height are then simulated during dynamic yawing motion, as well as using a different device type, once again in a stable attitude and a dynamic one. A new simulation environment, the Semi-analytic Tool for End-of-Life Analysis software (STELA), is introduced and briefly discussed. The same satellites are then simulated in STELA for further comparison.

Chapter 6: Results Discussion

This section discusses the results obtained from the measurements and simulations in more detail. The hardware-related outcomes are covered in the beginning, comparing the atmospheric test to the vacuum test. The section ends with the explanation of the various MATLAB simulations and the comparison thereof to those from STELA.

Chapter 7: Conclusions and Recommendations

The project closes with a conclusion drawn up from the collected results, and any recommendations are provided for future research on solutions to the problem.

Appendices

There are four appendices which serve as aids to various sections in the body of the thesis. They contain the Atmel Studio code, EAGLE schematics, MATLAB code and Simulink blocks used in this study.

Chapter 2

Literature Study

CubeSats have recently become very popular amongst space organisations, universities and private companies alike. There have been over 1000 CubeSats launched from 2003 to 2018, and with more than 80% of them launched after 2014, the trend shows that this number will continue increasing [1], [2]. The fundamentals of CubeSats will be discussed in this chapter, followed by the impact they have on the space environment as well as ways to mitigate or eliminate this impact.

2.1 CubeSat Satellites

CubeSats are a type of small satellite, classed as nanosatellites, that follow a specific architecture or standard. The CubeSat Design Specification was first developed in 1999 by Stanford University's Professor Bob Twiggs and California Polytechnic State University's Professor Jordi Puig-Suari [3]. It defines one unit of the satellite (1U) to be a cube with 10 cm side lengths and a weight of no more than 1.33 kg [4]. This standard was created for educational opportunities, allowing university students to get the first-hand experience in designing, building and controlling satellites.

The CubeSat Design Specification has expanded since its inception to include larger sized units; mostly due to the increased use in commercial space. The units can be stacked together to form many larger CubeSats with varying form factors, such as 2U and 3U, or even 27U. Each unit keeps its original dimensions specified by the architecture, and thus a 3U CubeSat can weigh up to 4 kg. Typical weight values for nanosatellites are in the range of 1 kg to 10 kg, which means that most 8U and larger CubeSats would be classified as microsatellites instead [5]. Figure 2.1 compares CubeSats from 1U up to 12U in size.

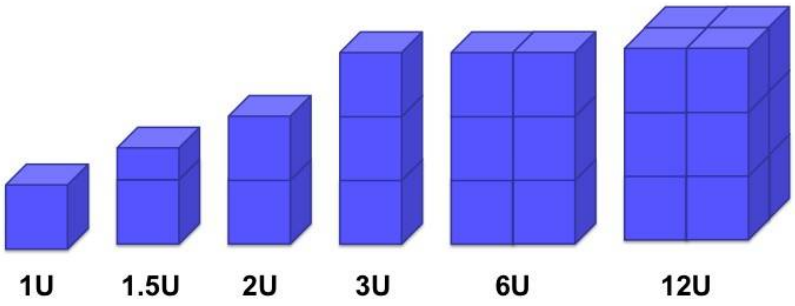
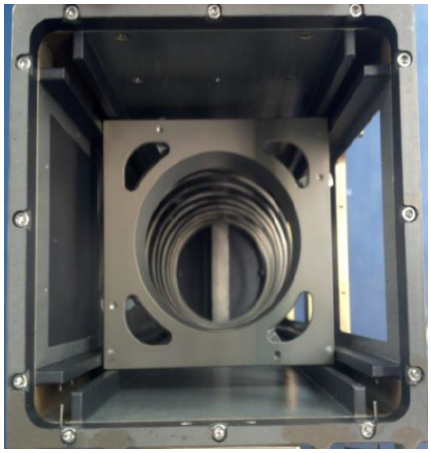


Figure 2.1: A Comparison of CubeSat Sizes [6]

Smaller satellites cost less to launch, as launch cost is usually based on weight. Satellite launches are most commonly dictated by the primary payload, usually the largest satellite on any particular launch vehicle, and the launch vehicle's capacity is sized accordingly. To maximise return for the launch operator, the launch vehicle's excess mass capacity is made available for smaller satellites to be launched as secondary payloads. These piggy-back launches, as they became known colloquially, are cost-effective for smaller satellite owners since they don't have to pay for a dedicated launch. On the other hand, they have to abide by the launch schedule and orbital requirements of the primary payload. California Polytechnic State University developed the Poly Picosatellite Orbital Deployer (P-POD) to deploy the CubeSats from the launch vehicle, shown in Figure 2.2 (a) and (b). The housing has rails at four edges to support the CubeSat during launch. CubeSats are in a class of containerised satellites and the P-POD and subsequent designs of CubeSat launch dispensers are one of the primary design drivers for the CubeSat standard. The launch dispenser allows for CubeSats to be launched as secondary payloads without increasing risk for the primary payload owner since during launch it is completely encapsulated in a flight-proven launch dispenser, thereby protecting the primary payload and other secondary payloads from potentially poorly built student satellites. The P-POD launch dispenser also provides simple integration with its standardised interface to the launch vehicle. The original design allows three 1U CubeSats or a single 3U CubeSat per dispenser. The P-POD will deploy the CubeSat(s) once the primary payload is safely released and no collisions will occur [1], [7].



(a) P-POD Cross Section [7]

(b) P-POD Outside [8]

Figure 2.2: P-POD Deployer

The original P-POD design has been modernised with the increased usage of 6U and larger CubeSats, and another dispenser, known as the Canisterized Satellite Dispenser (CSD), has been developed. This modernisation has created a second class of CubeSat dispensers, which consists of tabs keeping the payload in place instead of rails, on a dispenser developed by Planetary Systems Corporation. The CSD enables payloads of up to 27U to be launched and deployed [5], [9]. A 3U sized CSD is shown in Figure 2.3, with a focus on the tab mechanism.

Orbits are classified by their height above the Earth's surface. CubeSats are primarily launched in Low Earth Orbit (LEO), which is the area of space up to 3000 km but is mostly referred to as orbit heights of less than 900 km [10]. It is rare to launch CubeSats higher than LEO; at time of printing, no CubeSats have been placed in Geosynchronous

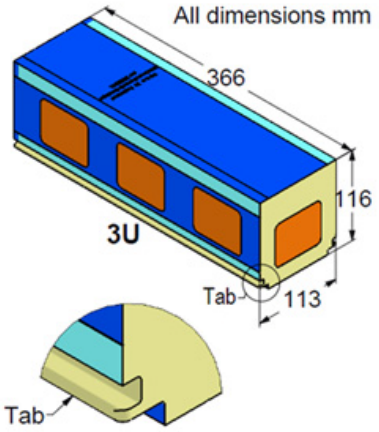


Figure 2.3: CSD System [9] (modified)

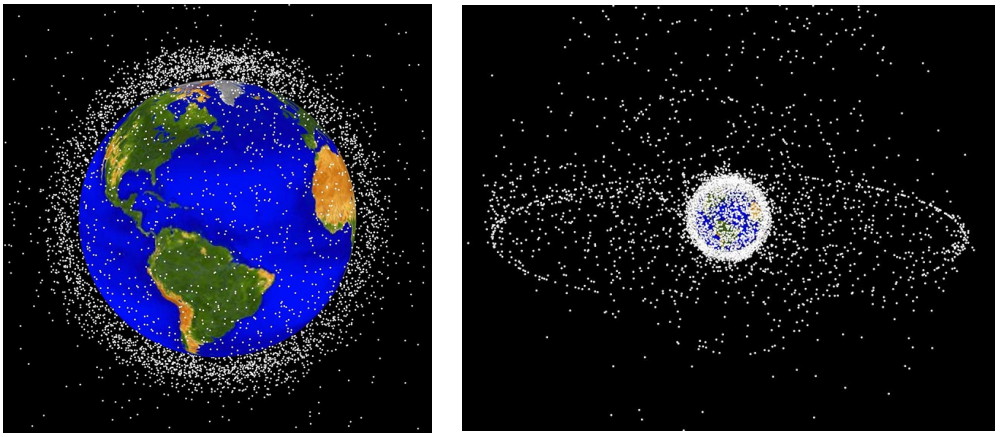
Orbit (GEO). The first CubeSat to be launched into GEO is currently planned for 2021. Only two CubeSats have been launched as interplanetary missions [11]. The twin 6U Mars Cube One (MarCO) CubeSats, named MarCO-A and MarCO-B, were initially scheduled to launch in March of 2016, but ultimately left LEO for Mars for a flyby in May of 2018 [12]. The two CubeSats' missions were to assist the InSight Mars lander during the landing phase if the mission, serving as communication relays of the flight information to the Mars Reconnaissance Orbiter [13]. They made their closest approach to Mars on the 26th of November 2018, during the InSight landing, yet only lasted roughly another month before the National Aeronautics and Space Administration (NASA) stopped receiving communication signals from the twin CubeSats. MarCO-B fell silent on the 29th of December 2018 while MarCO-A lasted nearly another week before sending its last correspondence on the 4th of January 2019 [14]. They had mission lifetimes of less than a year each, which is not uncommon for CubeSats, especially considering that these survived in interplanetary space.

The short mission lifespan of CubeSats is a consequence of their sizes. Commercial off-the-shelf (COTS) electronics became widely used amongst CubeSat manufacturers for miniaturised versions of the payloads and subsystems found in larger satellites. These COTS components help to keep the costs low, but also cause their short mission lifetimes, as they are not space class, radiation hardened components. The small mass and volume budgets also do not allow for radiation shielding to be included. While the small, low-cost design means that the electronics are more susceptible to radiation, it does, however, easily allow large numbers of CubeSats to be deployed to form satellite constellations [15].

2.2 Orbital Debris

Orbital debris, space debris or even space junk are terms used to describe any debris that orbits the Earth. Since Sputnik 1 in 1957, thousands of satellites have been launched into space in LEO, GEO and further. While some LEO debris may fall back down to Earth and either burn up in the atmosphere or fall somewhere on the planet, many other objects are still orbiting. The Inter-Agency Space Debris Coordination Committee defines orbital debris as "...all man made objects including fragments and elements thereof, in Earth orbit or re-entering the atmosphere, that are non functional" [16]. This definition shows that orbital debris not only consists of whole non-functioning satellites but any fragments thereof caused in collisions or explosions from rocket upper-stages. Most debris is found

in LEO [17]; consequently, LEO is the area of focus in this study because of CubeSats mostly being injected into LEO, as mentioned in Section 2.1. Figure 2.4 consists of two computer-generated images that show the objects with a diameter larger than 10 cm currently being tracked in (a) LEO and (b) GEO, with approximately 95% of these objects being inoperable satellites and fragments, or debris.



(a) Objects in LEO (b) Objects in GEO

Figure 2.4: Objects Tracked in Orbit (debris not to scale) [18]

The highest concentration of LEO debris is found between 750 km and 1000 km [18]. This flux of orbital debris in LEO, which is now more substantial than the meteoroid field in the same region, poses a severe risk to functional satellites and even the International Space Station (ISS) in the form of collisions. Even with no new satellite launches, collisions in LEO will continue for at least the next 200 years. [19]. In a kind of continuous cycle, collisions - whether caused by debris or not - contribute greatly to the formation of more orbital debris, as the two colliding objects form many fragments during the process. This debris formation from crashes can cause what is known as the Kessler Effect, where collisions have a snowball effect in creating more debris until a debris belt is orbiting the Earth. No satellites can be utilised in the region afterwards [20]. Collisions are mostly accidental, but there have been cases where they were deliberate. In 2007, the Chinese intentionally shot down their Fengyun-1C weather satellite which contributed significantly to the orbital debris. An accidental collision occurred between the working American Iridium 33 communications satellite and the decommissioned Russian Kosmos-2251 communications satellite in 2009. These two events caused such large amounts of orbital debris that it now makes up about a third of all indexed debris [18]. Collisions are, however, not the only cause of orbital debris; another being the previously mentioned defunct satellites and upper launch stages that have been abandoned in orbit and now classified as orbital debris.

It is challenging to track and monitor all the debris because most of the debris is not that of the intact satellites and rocket stages. There are over 23 000 known orbital debris objects that are 10 cm in size or larger [18]. The United States (the US Space Surveillance Network), the Russian Federation, Japan and Germany track them using radar. The US, the Russian Federation, Japan France, the United Kingdom and Switzerland follow these debris fragments using telescopes [21]. However, there are an estimated 500 000 debris objects between 1 cm and 10 cm while the estimated number of objects larger than 1 mm up to 1 cm surpasses 100 million [18]. The Long-Duration Exposure Facility (LDEF) was

a satellite sent to LEO and returned to Earth after more than five years. The number of collisions with small debris particles were then counted to estimate the amount of debris in the region. There were more than 30 000 craters from collisions, and sub-millimetre sized particles made all of them [21]. The LDEF results show that debris is plentiful in even smaller sizes than expected, showing the enormous range of debris sizes.

Manoeuvres can be made by satellites and the ISS to avoid collisions when the probability thereof is high enough and if the object is noticed with an adequate amount of time to manoeuvre. The National Aeronautics and Space Administration (NASA) uses the following guideline to move to prevent debris collisions with the space shuttle - now out of commission - and the ISS respectively [22]:

"1 Probability > 1 in 100,000: Maneuver (sic) if it will not result in significant impact to mission objectives.

2 Probability > 1 in 10,000: Maneuver (sic) unless it will result in additional risk to crew (reflight, additional spacewalk, etc.)."

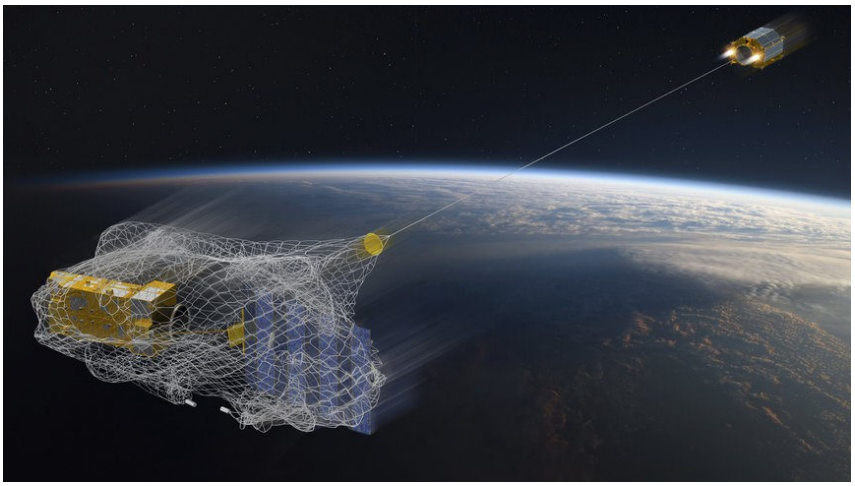
An ISS manoeuvre happens roughly once a year [18]. While manoeuvres can avoid most collisions, it is often a waste of propulsion or time, sometimes delaying missions until a later point in time. Long-term solutions of orbital debris mitigation have been set up by the United Nations (UN), and they include specifications to the prevention of the creation of more debris and the removal of current debris. The guideline states that satellites must deorbit within 25 years of the satellite's end of mission or EoL [21]. CubeSats have to adhere to this standard strictly, and a debris mitigation plan should be submitted before a CubeSat can be launched [23]. There are two types of strategies that can be implemented to follow these guidelines, namely active and passive debris removal, each with various methods of achieving the set regulations. They will be looked into in more depth in the following two sections.

2.2.1 Active Debris Removal

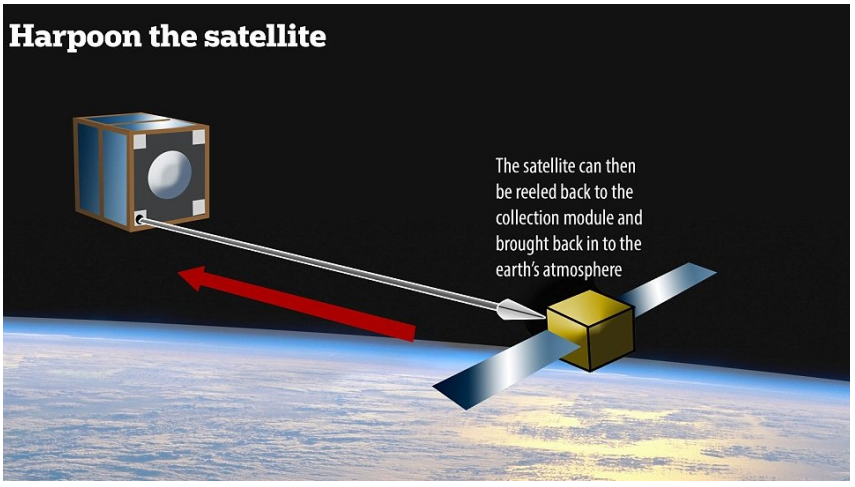
Currently, the only way to remove orbital debris that is already in orbit is by using active debris removal (ADR). ADR is comparable to garbage removal on Earth, where the debris is actively collected and either brought back to Earth (usually for LEO debris) or moved into a higher disposal orbit (GEO debris and higher). A dedicated satellite would be sent into orbit and then later brought back to Earth with the captured debris. The collection method could either be done by using a net or a harpoon of sorts to capture and reel in the debris. Concepts proposed by the European Space Agency (ESA) Clean Space Office's plan to implement ADR using a net and the University of Surrey's ADR plan involving a harpoon is shown in Figure 2.5. These are both known as collective ADR methods - the first of six ADR methods to be considered in this subsection - and are mostly used for large items of debris [24], [25].

The second type of ADR is the laser-based method. This method would use either a ground-based or space-based high-power pulsed laser systems, or even combinations of the two, to shoot plasma jets at the debris which would slow them down and perturb their orbits. Laser ablation is capable of removing both large and small debris fragments as well as those that may be tumbling or spinning. The laser could also be light-based, which would use photon pressure to disturb the debris object [28].

The ion-beam shepherd-based method is the third kind of ADR technique proposed. It utilises the opposite logic of laser-based methods by emitting a beam of quasi-neutral



(a) ADR Using a Net



(b) ADR Using a Harpoon

Figure 2.5: Collective ADR Concepts [26], [27]

plasma towards the debris object to supply it with a propulsive force. In short, the ions would effectively push the debris, which requires more power than the laser-based method and cannot be done from the Earth as it can only work in close-range situations. The ion beam would, therefore, be placed on a debris chaser satellite known as the shepherd satellite, fitted with a propulsion system to stabilise the shepherd during ion beam radiation [24].

The fourth ADR method to be considered is the tether-based method. This solution proposes using a wire electrodynamic tether (EDT) which is exposed at the anode end, attached to a large satellite. The tether would magnetically attract any small debris fragments and slow them down to enable deorbiting. The fragments attracted would be small enough to burn in the atmosphere upon re-entry. The tether does, however, have to be a few kilometres long for it to work. It does not require any power but is slow in its operation [24], [29].

The fifth ADR technique that can be employed to deorbit a satellite is by using propulsion to deorbit the satellite. Unlike the other methods of ADR, this does not involve another satellite deorbiting the debris. It is similar to passive debris removal in that the thrusters already have to be on the satellite before launch. Two thrusters are placed on opposite sides of the satellite, with one used to accelerate the satellite, and the

other used to decelerate it. The one for deceleration would be the primary thruster for deorbiting the satellite, reducing the orbital energy and decreasing the altitude [30].

The final method of ADR, known as a satellite-based method, proposes the use of microsatellites to remove debris. The satellites would be deployed to attach a passive deorbiting device, such as an EDT, to the debris objects themselves. A robotic arm would be required on the satellites to accomplish this. The debris would then passively deorbit. Satellite-base ADR is the most complex ADR method compared to the previous four methods mentioned [24], [25].

A summarised version of the six ADR methods is given in Table 2.1

Table 2.1: ADR Methods Summarised

Method	Description	Pros	Cons
Collection-Based	Collect the debris using a net or harpoon.	Various collection-based methods exist.	Harpoon not good for tumbling satellites.
Laser-Based	Ablate the debris with a laser to slow it down.	Feasible and low cost.	Angle and range of operation is limited.
Ion-Beam Shepherd-Based	Shoot ion beams at debris to accelerate it.	Quicker than laser, also feasible.	Cannot operate from Earth, requires more power than laser.
Tether-Based	Use an electromagnetic tether attached to a large satellite to magnetically attract the debris using the Earth's magnetic field to slow it down.	No power nor maintenance is required.	Slow to deorbit.
Propulsion-Based	Use propulsion to decelerate the satellite.	Low cost and complexity.	Thrusters have to already be on satellite.
Satellite-Based	Use a satellite with a robotic arm to attach a passive deorbiting device onto the debris.	Actual deorbit device placed on debris requires no power nor maintenance.	High complexity, will not work for tumbling debris. Slow once device is attached.

2.2.2 Passive Debris Removal

The focus and goal of passive debris removal (PDR) is to prevent more orbital debris forming. It is not used to remove any current debris in orbit. At a satellite's EoL a device of sorts would get deployed to serve as an aid in the deorbiting process. The device can either be an EDT as (mentioned in section 2.2.1) which slows the satellite using the Earth's magnetic field or be a drag-based device. The drag-based device increases the total area of the satellite without increasing its mass to increase its susceptibility to atmospheric drag, to decrease the orbit's energy and altitude [10], [31]. Figure 2.6 depicts a sail used to deorbit a satellite, also by the University of Surrey.

The difference between ADR and PDR is that with PDR nothing is actively done to the satellite once declared debris, besides deploying its EoL deorbit device. The satellite is also deorbited in an uncontrolled manner, and therefore should only be done on smaller satellites as precision re-entering of the atmosphere cannot be done. PDR, therefore, requires a power supply independent to its own to deploy the device if the satellite dies at EoL. The device also needs to be autonomous as it must be able to detect that the satellite is no longer functional. This can be achieved by having the satellite send a message to the device's on-board computer or microcontroller once every month for example, and when the message is not received after a few times, the device must be deployed. The deorbit

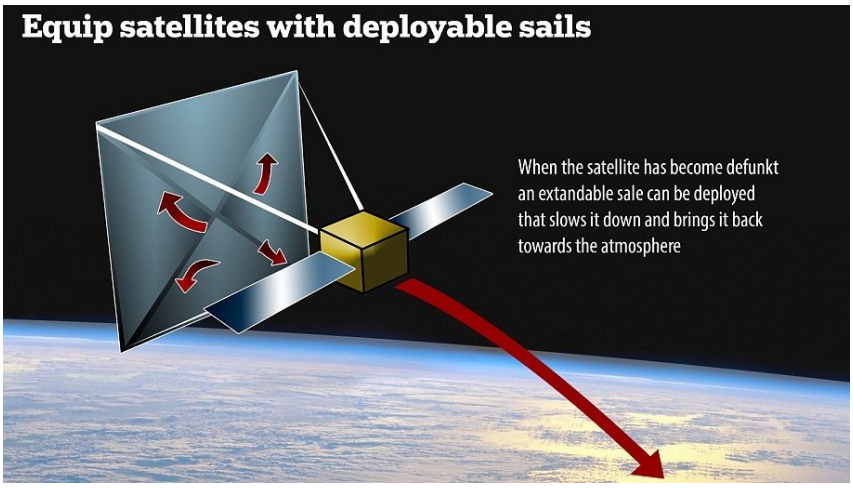


Figure 2.6: PDR Using a Sail [27]

device can also be triggered manually if the satellite is still operational, but at the end of its mission or if it no longer required.

The method of using a tether was discussed previously. To passively deorbit using drag, a sail, or something similar, would have to be deployed. The concept of solar sailing, which utilises the energy of the Sun for propulsion, brought about the idea of using a sail to deorbit a satellite. Solar sailing uses photons from the Sun on a thin, light almost-perfect reflector sail to propel the satellite forward. The same concept can be used for deorbiting, just in a sense that is more similar to sails on a ship; i.e. use the atmosphere. The deorbit sail would not propel the satellite forward though, it would use the sail to remove orbit energy and lower the orbit altitude, causing it to deorbit over time. The sail would still need to be light and could still be reflective to use solar sailing to augment the deorbiting even further [24], [25].

Similar to a deorbit sail device is the deorbit balloon device. The balloon would function in the same way at the sail, just using a gas such as oxygen or nitrogen to inflate and then deorbit with atmospheric drag. The balloon can be made of the same material as a solar sail to once again use solar radiation pressure (SRP) to aid in the deorbiting. The balloon can be made of two circular pieces of Mylar or similar materials used for solar sailing, which will then be inflated at the satellite's EoL to form a spheroid. It is simpler than using a sail which would require booms to be deployed before the sail unfolds, and the amount of gas required is minuscule compared to what is needed in atmospheric conditions. However, a sail can still work if it collides with a small debris fragment, where the balloon would leak and can no longer aid in deorbiting. It has been suggested to use both an EDT and drag-based deorbiting together to lower the risk of a failed deorbit [24].

Chapter 3

Theoretical Background

This chapter presents the fundamental concepts which include definitions of satellite motion utilising a two-body problem and orbit parameters used. Coordinate systems used to describe the position and attitude of a satellite are looked into, specifically the three main coordinate systems used. Thereafter ways that an orbit is perturbed are investigated, starting with atmospheric drag, as well as specific atmospheric models that can be used when working with atmospheric density. The last perturbations to be studied are the effects of gravity and the Earth's oblateness and solar radiation pressure. The theoretical knowledge from this chapter is utilised in the MATLAB simulations for the deorbiting device, to be discussed in Chapter 4.2.

3.1 Orbits and Orbital Dynamics

3.1.1 Description of Satellite Motion

A classical two-body problem is necessary to understand how satellites orbit the Earth and introduce the essential theory and orbital mechanics concepts required for orbit determination and propagation. The two-body problem relies on the physics of Isaac Newton and Johannes Kepler's laws. Kepler used observations and data collected by scientists in the 16th and 17th centuries to derive the rules for planetary motion. Newton was able to justify Kepler's rules for planetary motion using his laws of mechanics and gravitation theory and as such derived Kepler's three laws of planetary motion, which are quoted as the following [10]:

"1 If two objects in space interact gravitationally, each will describe an orbit that is a conic section with the centre of mass at one focus. If the bodies are permanently associated, their orbits will be ellipses; if not, their orbits will be hyperbolas.

2 If two objects in space interact gravitationally (whether or not they move in closed elliptical orbits), a line joining them sweeps out equal areas in equal intervals of time.

3 If two objects in space revolve around each other due to their mutual gravitational attraction, the sum of their masses multiplied by the square of their period of mutual revolution is proportional to the cube of the mean distance between them."

The second law is illustrated in Figure 3.1, where $t_2 - t_1 = t_4 - t_3$ and $A_1 = A_2$, while the third law is mathematically expressed as follows [10]:

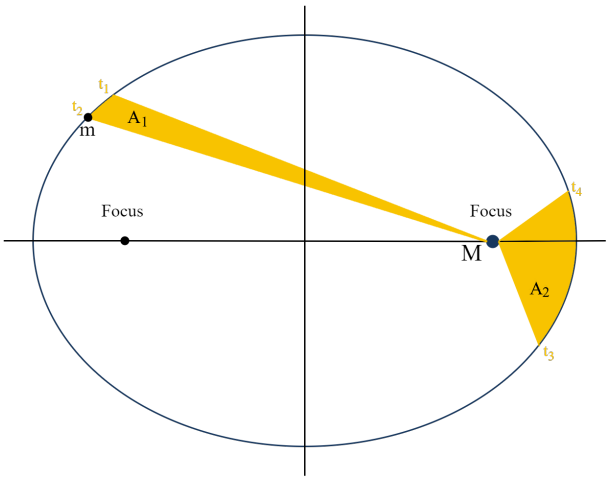


Figure 3.1: Kepler's Second Law of Planetary Motion

$$(m+M)P^2 = \frac{4\pi^2}{G}a^3 \tag{3.1}$$

where m and M are the two masses - the more massive one denoted with the capital letter - P is their mutual period of revolution, G is Newton's gravitational constant and a is the mean distance between the objects.

These laws lead to the two-body problem and the forces acting upon them. The state equations are split into two decoupled one-body problems, resulting in independent sets of state equations. The two-body problem has two points, P_0 and P_1, with constant masses of M and m respectively. They are placed in arbitrary positions in an inertial frame with origin O. This is depicted in Figure 3.2.

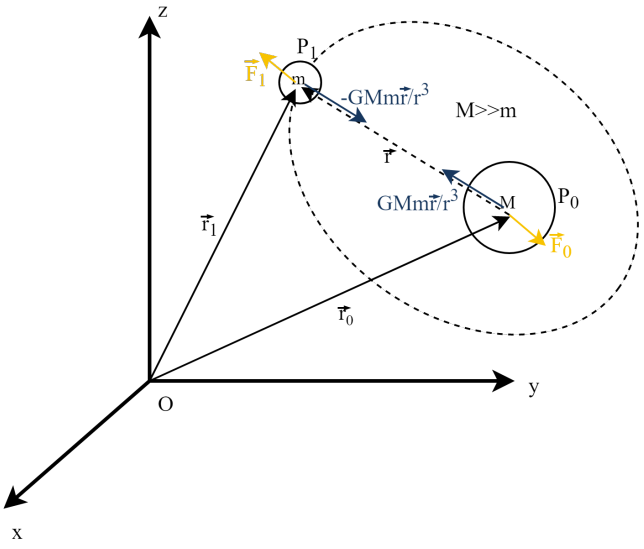


Figure 3.2: Two-Body Problem Geometry and Forces

The relative point position is given by

$$\vec{r} = \vec{r}_1 - \vec{r}_0 \tag{3.2}$$

with $\vec{r_0}$ the position of P_0 to the origin and $\vec{r_1}$ that of P_1. Using Newton's second law and the law of universal gravitation yields the following equations of motion [32]:

$$\begin{aligned} \dot{\vec{r}}_0 &= \vec{v}_0 \\ \dot{\vec{r}}_1 &= \vec{v}_1 \end{aligned} \tag{3.3}$$

and

$$\begin{aligned} \dot{\vec{v}}_0 &= \frac{GMm}{Mr^3}\vec{r} + \frac{1}{M}\vec{F}_0 \\ \dot{\vec{v}}_1 &= -\frac{GMm}{mr^3}\vec{r} + \frac{1}{m}\vec{F}_1 \end{aligned} \tag{3.4}$$

The first equations in both sets are known as the centre of mass (CoM) equations, and the last two are the relative body equation. Newton's law of gravitation gives the gravity force between P_0 and P_1:

$$\vec{g}_{01} = \frac{GMm\vec{r}}{r^3} \tag{3.5}$$

Applying these laws to the relative point position and velocity yields [32]:

$$\dot{\vec{r}} = \vec{v} \tag{3.6}$$

and

$$\begin{aligned} \dot{\vec{v}} &= -\frac{GMm}{mr^3}\vec{r} + \frac{1}{m}\vec{F}_1 - \frac{GMm}{Mr^3}\vec{r} - \frac{1}{M}\vec{F}_0 \\ &= -\frac{G(M+m)}{r^3}\vec{r} + \frac{1}{m}\vec{F}_1 - \frac{1}{M}\vec{F}_0 \\ &= -\frac{G(M+m)}{r^3}\vec{r} + \frac{1}{m}\left(\vec{F}_1 - \frac{m}{M}\vec{F}_0\right) \end{aligned} \tag{3.7}$$

The forces, F_0 and F_1, are external forces which account for perturbing forces to be studied later on. If these external forces are equal to zero, the equations form the *free response* of the system, which has a closed-form. If these forces are assumed to be much smaller than gravity, i.e. $|\vec{F}_0|, |\vec{F}_1| \ll |\vec{g}_{01}| = \frac{GMm}{r^2}$, it becomes known as the forced response which can be approximated as a perturbation of the free-response. In the free-response, the equations split into two non-interacting equations because the external forces are set to zero. With mass M much larger than mass m, the relative body equation has a non-trivial solution which is known as the restricted two-body problem. This restricted two-body problem was found by Newton and obeyed Kepler's laws for a closed orbit. The position of the CoM denoted $\vec{r}_c$ can be found with [32]:

$$\vec{r}_c = \frac{M}{M+m}\vec{r}_0 + \frac{m}{M+m}\vec{r}_1 \tag{3.8}$$

If equation 3.8 is applied to equations 3.3 and 3.4, the following state equations are found:

$$\dot{\vec{r}}_c = \vec{v}_c \tag{3.9}$$

and

$$\dot{\vec{v}}_c = \frac{m}{M}\frac{\vec{F}_0 + \vec{F}_1}{1 + \frac{m}{M}} \tag{3.10}$$

This CoM equation and the one in equation 3.6 show that the gravity force between P_0 and P_1 is internal, which means it does not influence the two-body CoM. To decouple the system, such that the forces are non-interacting, one has to assume M is much larger than m, such that $\frac{M}{m} \to \infty$, which yields the following from equations 3.7 and 3.10:

$$\begin{aligned} \dot{\vec{r}}_c &= \vec{v}_c \\ \dot{\vec{r}} &= \vec{v} \end{aligned} \tag{3.11}$$

and

$$\begin{aligned} \dot{\vec{v}}_c &= 0 \\ \dot{\vec{v}} &= -\frac{GM\vec{r}}{r^3} + \frac{\vec{F}_1}{m} \\ &= -\frac{\mu\vec{r}}{r^3} + \frac{\vec{F}_1}{m} \end{aligned} \tag{3.12}$$

where $\mu = GM$. The last two equations in 3.11 and 3.12 together amount to the restricted two-body equation.

$$\dot{\vec{v}} = -\frac{\mu\vec{r}}{r^3} \tag{3.13}$$

If there are no external forces, i.e. there are no perturbations, the last term is dropped in the second equation, resulting in equation 3.13. Orbits that obey the free-response (when gravity is the only force) are known as Keplerian orbits [10].

To get to orbit parameters used, one has to place the two-body problem in inertial coordinates. If the two-body inertial plane is in the Earth-centred inertial (ECI) frame, the origin point becomes $O = E$, where E is the centre of the Earth. Kepler's first law leads to what is known as the first conservation law, where the free-response lies in a plane known as the orbital plane. The first conservation law is mathematically proven as follows, given that the acceleration $\dot{\vec{v}}$ is always opposite to $\vec{r}$ [32]:

$$\dot{\vec{h}} = \dot{\vec{v}} \times \vec{v} + \vec{r} \times \dot{\vec{v}} = \vec{r} \times \dot{\vec{v}} \tag{3.14}$$

Replacing $\dot{\vec{v}}$ above with that from equation 3.13 leads to:

$$\dot{\vec{h}} = \vec{r} \times -\frac{\mu\vec{r}}{r^3} = 0 \tag{3.15}$$

Thus, the angular momentum $\vec{h}$ is the first constant of motion, or a constant vector, in the direction of the orbital pole. A constant vector in direction is understood as inertial, and in turn, the orbital plane orthogonal to $\vec{h}$ and is inertial. As a result, all three inertial coordinates in $\vec{h}$ are constant.

Therefore, to validly use the free response two-body equation, the following quoted assumptions need to be made [31]:

"1 The mass of the satellite is negligible compared to that of the attracting body. This is reasonable for artificial satellites in the foreseeable future.

2 The coordinate system chosen for a particular problem is inertial. ...It removes derivatives of the coordinate system itself when differentiating vectors. ...

3 The bodies of the satellite and attracting body are spherically symmetrical, with uniform density. This allows us to treat each as a point mass.

4 No other forces act on the system except for gravitational forces that act along a line joining the centers (sic) of the two bodies."

3.1.2 Orbit Parameters

In order to specify a Keplerian orbit, the following information is required [10]:

1. The shape and size of the orbit.
2. The plane's orientation relative to the equator and celestial pole.
3. How the semi-major axis is oriented in the plane.
4. The satellite's location in orbit.

The classical orbital elements imply a solution to the equations 3.11 and 3.13, represented by a vector given as

$$\vec{p}(\vec{r},\vec{v}) = \text{function of } [\Omega, i, a, e, \omega, \nu] \tag{3.16}$$

In order, these elements are the right ascension of the ascending node (RAAN), the orbital inclination, the semi-major axis, the eccentricity, the argument of perigee and the true anomaly [32]. One solution for the free response two-body equation using most of these orbital elements is given as [33]:

$$r = \frac{a(1-e^2)}{1+e\cos\nu} \tag{3.17}$$

To define RAAN, one has to define the ascending node and the line of nodes. The ascending and descending nodes are the points where the orbit crosses the equatorial plane. The ascending node is where the orbit crosses the equatorial plane going from the southern hemisphere to the northern hemisphere. If the axes directions in ECI are labelled $\vec{x}_i$, $\vec{y}_i$ and $\vec{z}_i$, the celestial north pole would be in the direction of $\vec{z}_i$. The line of nodes is defined as the entire line where the orbital and equatorial planes intersect from the ascending node to the descending node. The RAAN can be defined as the angle between $\vec{x}_i$ the direction of the Northern spring equinox - known as the vernal equinox - and the line of nodes [32]. The orbital inclination is the angle between the equatorial plane and the orbital plane. If satellites travel with the rotation of the Earth, known as a prograde orbit, the value for i will be between 0° and 90°. Otherwise, if the inclination is between 90° and 180°, it is known as retrograde orbit [10]. These two orbital elements are calculated with the first conservation law which states that the orbital plane is inertial, and are the two elements to describe the orientation of the orbital plane. Figure 3.3 serves as a visual aid to help define the two elements.

The next two orbital elements are used to describe the orbit's shape and size. The semi-major axis is the distance from the centre of the orbit to the furthest point along the long axis, known as the line of apsides. The point on this axis that is closest to Earth

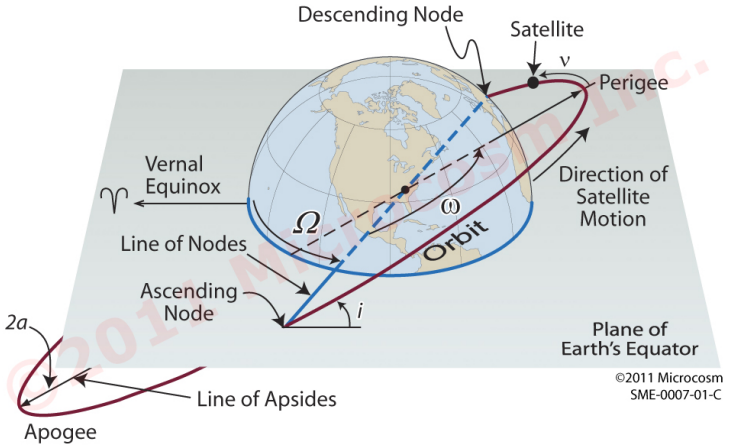


Figure 3.3: Keplerian Orbit Parameters [10]

(the more massive body of mass) is known as the perigee, while the furthest point is the apogee. The semi-major axis can be found using the radius of the Earth and the perigee and apogee heights as follows [10]:

$$\begin{aligned} a &= \frac{R_E + h_a}{2} + \frac{R_E + h_p}{2} \\ &= R_E + \frac{h_a + h_p}{2} \end{aligned} \tag{3.18}$$

where R_E is the equatorial radius of the Earth and h_a and h_p are the apogee height and perigee height respectively. Figure 3.4 shows these definitions.

The eccentricity is a measure of how elliptic an orbit is and is always a value between 0 and 1, with zero a perfect circular orbit and the closer to one the more elliptical the orbit. At $e = 1$ the orbit becomes a parabola, and if larger than one the orbit trajectory forms a hyperbola. These open-form orbits are used to leave the Earth's orbit and typically used for interplanetary missions. Both the closed-form and open-form orbits are considered conic sections [10]. It is calculated using the ratio of the distance between the centre of the Earth to the centre of the orbit and the semi-major axis, therefore [33]:

$$e = \frac{c}{a} \tag{3.19}$$

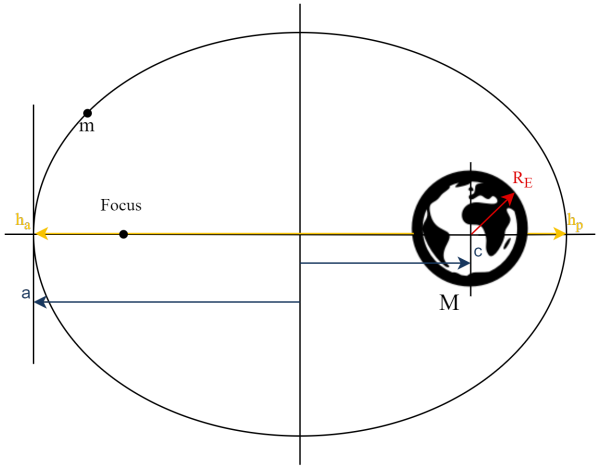


Figure 3.4: Definition of Semi-Major Axis, Apogee Height and Perigee Height

where c is the distance from the Earth's centre to the centre point of the orbit, given as $c = a - (R_E + h_p)$, also shown in Figure 3.4. The orbital plane eccentricity vector is given as [32]:

$$\vec{e} = \frac{\vec{v} \times \vec{h}}{\mu} - \frac{\vec{r}}{r} \tag{3.20}$$

The fifth orbital element, the argument of perigee ω, is used to represent the rotational orientation of the semi-major axis. It is represented by the angle between the ascending node and the perigee point. Therefore, an argument of perigee of 0° means that the ascending node coincides with the perigee while the descending node is at the apogee point. The argument of perigee is also illustrated in Figure 3.3. The final orbital parameter is the true anomaly. It is used to define the position of the satellite within orbit, and is, therefore, not constant but a function of time. The true anomaly is the angle from the perigee to the satellite.

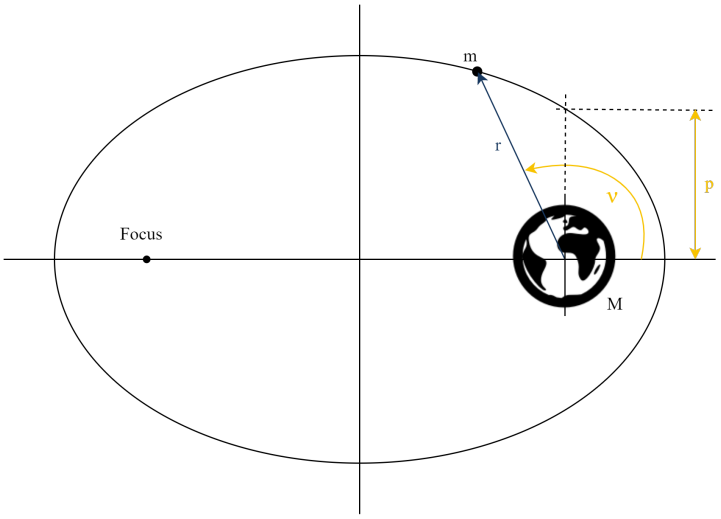


Figure 3.5: Definition of the True Anomaly and the Semi-Latus Rectum

The planar curve can be found from the dot product of the radius vector and the eccentricity vector and is defined as [32]:

$$\begin{aligned} \vec{r} \cdot \vec{e} &= \vec{r} \cdot \left(\frac{\vec{v} \times \vec{h}}{\mu} - \frac{\vec{r}}{r}\right) \\ &= \frac{\vec{h} \cdot (\vec{r} \times \vec{v})}{\mu} - r \\ &= \frac{h^2}{\mu} - r \end{aligned} \tag{3.21}$$

The true anomaly is the argument of the radius vector with respect to the eccentricity vector in polar form, as such the expression for the radius results in:

$$\begin{aligned} \vec{r} \cdot \vec{e} &= re\cos\nu \\ \therefore re\cos\nu &= \frac{h^2}{\mu} - r \\ \therefore r &= \frac{\frac{h^2}{\mu}}{1 + e\cos\nu} \end{aligned} \tag{3.22}$$

The ratio of h^2 to μ from equation 3.21 is known as the semi-latus rectum or parameter of the ellipse, p. It is the distance orthogonal to the semi-major axis from the focus to the orbit, also shown in Figure 3.5. The expression to find the point p can also be found using the normal form of the ellipse equation and the definition $b = a\sqrt{1-e^2}$ of the semi-minor axis [32]:

$$\begin{aligned} \frac{c^2}{a^2} + \frac{p^2}{b^2} &= 1 \\ \therefore e^2 + \frac{p^2}{a^2(1-e^2)} &= 1 \\ \therefore p &= a(1-e^2) \end{aligned} \tag{3.23}$$

Kepler's third law of planetary motion can further be verified by taking the definitions of p from equation 3.22 and combining it with equation 3.23 to find a constant angular momentum:

$$h = \sqrt{\mu p} = \sqrt{\mu a(1-e^2)} \tag{3.24}$$

Another important parameter of a satellite is its orbital velocity. The energy conservation law is used to find the orbital velocity. The law states that the sum of the potential and kinetic energy in the system must equal the total energy, or the specific mechanical energy, denoted with ε. This energy is constant due to the two-body restrictions and assumptions made, and is calculated as follows [32]:

$$\begin{aligned} E_{tot} = \varepsilon &= E_{potential} + E_{kinetic} \\ &= -\frac{m\mu}{r} + \frac{1}{2}mv^2 \\ &= \frac{v^2}{2} - \frac{\mu}{r} \end{aligned} \tag{3.25}$$

The masses can be eliminated because they are assumed constant. The angular momentum, at the point of perigee where $\nu = 0$ and $r = a - c = a(1-e)$, can be given as follows [32]:

$$\begin{aligned} h &= |\vec{r} \times \vec{v}| \\ &= rv \\ &= a(1-e)v \end{aligned} \tag{3.26}$$

The angular momentum is also constant, as proven in equation 3.15 and 3.24, and is, therefore, the same along the entire orbit. Rewriting equation 3.26 to have v as the subject and substituting it into equation 3.25 yields:

$$\varepsilon = \frac{h^2}{2a^2(1-e^2)} - \frac{\mu}{a(1-e)} \tag{3.27}$$

and from equation 3.24 we have that $h^2 = \mu a(1-e^2)$ resulting in:

$$\begin{aligned} \varepsilon &= \frac{\mu a(1-e^2)}{2a^2(1-e^2)} - \frac{\mu}{a(1-e)} \\ &= -\frac{\mu}{2a} \end{aligned} \tag{3.28}$$

The specific mechanical energy is therefore only dependent on the semi-major axis of the orbit, and the orbit velocity can be found from this and equation 3.25 as:

$$v = \sqrt{\frac{2\mu}{r} + 2\varepsilon} = \sqrt{\frac{2\mu}{r} - \frac{\mu}{a}} \tag{3.29}$$

Moreover, for a circular orbit where $r = a$:

$$v = \sqrt{\frac{\mu}{a}} \tag{3.30}$$

This equation is known as the Vis Viva equation, and it shows that the velocity of the satellite increases as the radius to the Earth decreases.

3.1.3 Coordinate Systems

How orbits work, that is the physics behind an orbit, has been discussed. It is now necessary to be able to describe the orientation of the satellite's body relative to a reference frame, defined as the attitude of a satellite in orbit. All space applications use coordinate systems with two defining properties, namely [10]:

1. The point of observation, known as the origin, which is the centre of the coordinate system.
2. The coordinate system's fixed direction or object.

This combination can result in a few coordinate systems, or reference frames, with the same origin point, but which are fixed about different directions or objects; or the other way around. Three coordinate systems will be considered in this section; namely, the ECI system mentioned briefly in sections 3.1.1 and 3.1.2, the orbit reference coordinates (ORC) system and the spacecraft body coordinates (SBC) system. Another essential concept is discussed after the explanation of the coordinate system, which is the transformation between the different systems.

Earth-Centred Inertial

The ECI system has the centre of the Earth at the origin point with the z_i-axis pointing to the celestial north pole. The x_i-axis is in the direction of the vernal equinox- the direction of the Sun's orbit when its orbit crosses the equator at the ascending node. The y_i-axis is orthogonal to both these axes, following the right-hand rule, as depicted in Figure 3.6. It is a fixed non-rotating coordinate system and is, for the most part, used for orbit analysis.

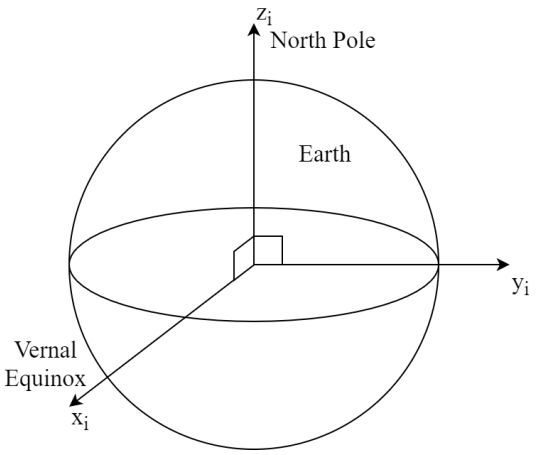


Figure 3.6: Earth-Centred Inertial Coordinate System

Orbit Reference Coordinates

The ORC frame is linked to the orbital path. In ORC the z_o-axis always points towards Nadir, which is the direction to the centre of the Earth. The x_o-axis points in the general direction of the satellite's movement while the y_o-axis once again is placed according to the right-hand set. It is a rotating coordinate system with the orbit position as the origin. Roll, pitch, yaw (RPY) is another name for this system as the RPY of the satellite is often how the attitude is described. The ORC system is shown in Figure 3.7.

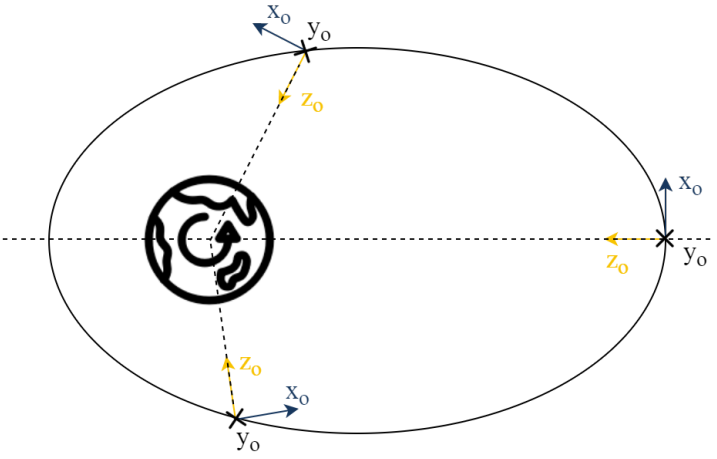


Figure 3.7: Orbit Reference Coordinates System

Spacecraft Body Coordinates

The SBC system is used to specify the direction of a specific component in the satellite and, therefore, relates to the physical body of the satellite. The origin point of this system is the satellite body itself with the direction of the axes chosen to be aligned with the specific component of interest. For this study, the x_b-axis is perpendicular to the deorbit sail - or balloon - in the general direction of motion, opposite to the direction of the force the drag would cause. The z_b and y_b-axes were chosen to represent the dimensions of the sail that would affect the drag force. As such, the z_b-axis is facing towards the satellite's zenith direction (upwards) to represent the length and the y_b-axis in the direction to

represent the width that completes the right-hand rule. This SBC system is represented in Figure 3.8.

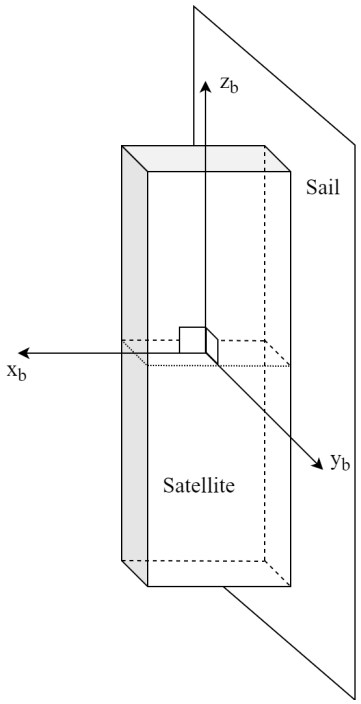


Figure 3.8: Spacecraft Body Coordinates System

3.1.4 Transformation Between Coordinate Systems

Each of these coordinate systems can also use various methods of representing the attitude of the satellite, for example, the RPY representation in the ORC system. Each unique coordinate system representation is mathematically correct and requires some transformation matrix to represent the attitude of the same satellite about coordinate systems. There are also three attitude representations used in this study to transform the orientation of one coordinate system to another, namely the direction cosine matric (DCM), Euler angles and quaternions.

Direction Cosine Matrix

The DCM is a rotation matric, which is a 3×3 orthogonal matrix defining relative rotations between two frames of reference. The DCM for ECI coordinates to the ORC frame is given as [33]:

$$\mathbf{A}_{I/O} = \begin{bmatrix} (\vec{r}_i \times (\vec{v}_i \times \vec{r}_i)^T \\ (\vec{v}_i \times \vec{r}_i)^T \\ (-\vec{r}_i)^T \end{bmatrix} \tag{3.31}$$

with the superscript T standing for the transpose of that element, and the position and velocity vectors are the unit vector versions thereof. Representing an attitude with a DCM does not produce any singularities, but can often have redundant parameters. Due to them being orthogonal, the transpose of a DCM is used to reverse the rotation of the axis transformation.

Euler Angles

Euler angles are the angles that represent the RPY of a satellite. It consists of three angles which is an efficient manner to represent attitude and is easily comprehended by humans; however, it can cause singularities when transforming. Using the Euler angles for transformation between coordinates systems can be done in twelve different ways, as they are non-commutative and must be done in certain orders to get to the correct attitude representation. These twelve types of three successive rotations can be organised into two main types, namely, type 1 using the so-called Cardan angles and type 2 using repeated axis parameterisation. Type 1 has a rotation around each different axis, whereas type 2 has repeat rotations around one axis. The Euler rotation sequence used for this study is the Euler 2-1-3 rotation, also known as the y-x-z rotation, illustrated in Figure 3.9. Roll, pitch and yaw are represented by φ, θ and ψ respectively.

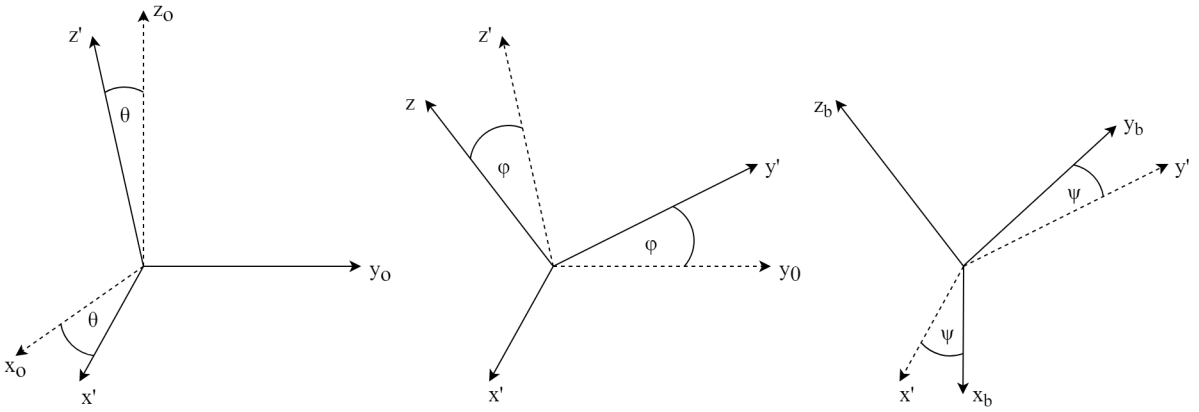


Figure 3.9: Euler 2-1-3 Rotation Sequence

The process in Figure 3.9 is mathematically represented as follows, with $\mathbf{A}$ the original orientation and $\mathbf{B}$ the final one:

$$\begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \mathbf{A}_{A'/A}(\theta) \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{3.32}$$

$$\begin{bmatrix} a_1'' \\ a_2'' \\ a_3'' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix} = \mathbf{A}_{A''/A'}(\varphi) \begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix} \tag{3.33}$$

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1'' \\ a_2'' \\ a_3'' \end{bmatrix} = \mathbf{A}_{B/A''}(\psi) \begin{bmatrix} a_1'' \\ a_2'' \\ a_3'' \end{bmatrix} \tag{3.34}$$

The combination of the previous three rotations forms a direction cosine matrix used to transform Euler 2-1-3 angles into a DCM. In this form, S and C are used as shorthand for the sin and cos functions, respectively.

$$\begin{aligned}\mathbf{A}_{B/A} &= \mathbf{A}_{B/A''}(\psi)\mathbf{A}_{A''/A'}(\varphi)\mathbf{A}_{A'/A}(\theta) \\ &= \begin{bmatrix} C\psi & S\psi & 0 \\ -S\psi & C\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\varphi & S\varphi \\ 0 & -S\varphi & C\varphi \end{bmatrix} \begin{bmatrix} C\theta & 0 & -S\theta \\ 0 & 1 & 0 \\ S\theta & 0 & C\theta \end{bmatrix} \\ &= \begin{bmatrix} C\psi C\theta + S\psi S\varphi S\theta & S\psi C\theta & -C\psi S\theta + S\psi S\varphi C\theta \\ -S\psi C\theta + C\psi S\varphi S\theta & C\psi C\theta & S\psi S\theta + C\psi S\varphi C\theta \\ C\varphi S\theta & -S\varphi & C\varphi C\theta \end{bmatrix}\end{aligned} \tag{3.35}$$

When this DCM is used to transform vectors in the ORC frame to the SBC frame, the superscripts change so that this DCM is defined as $\mathbf{A}_{O/B}$. From this DCM the RPY can be extracted as follows:

$$\begin{aligned}\theta &= \arctan 2(\mathbf{A}_{31}, \mathbf{A}_{33}) \\ \varphi &= -\arcsin(\mathbf{A}_{32}) \\ \psi &= \arctan 2(\mathbf{A}_{12}, \mathbf{A}_{22})\end{aligned} \tag{3.36}$$

where $\arctan 2$ is the four-quadrant inverse tangent function.

Quaternions

Another type of attitude representation is the quaternion, $\vec{q}$, which is an extension to the complex number system, comprised of four elements which are sum of an imaginary vector and a weighting scalar value. The quaternions also yield no singularities, unlike the Euler angles when, say, $\varphi = \frac{\pi}{2} rad$. Using trigonometric functions, like in the Euler angle representation, is computationally expensive. These functions increased the complexity and thus, the amount of time to successfully calculated. Quaternions are popular during numerical computations due to be these two characteristics [32]. A quaternion is represented as follows:

$$\vec{q} = q_1 i + q_2 j + q_3 k + q_4 \tag{3.37}$$

Quaternions were invented by the Irish mathematician William Hamilton in 1843 and are reliant on of Euler's theorem, which states that a rigid body with one fixed point's finite rotation can be expressed by a single rotation about some fixed axis [32], [33]. This fixed axis is known as the Euler-axis and can be represented by a unit vector in the ORC system, as illustrated in Figure 3.10.

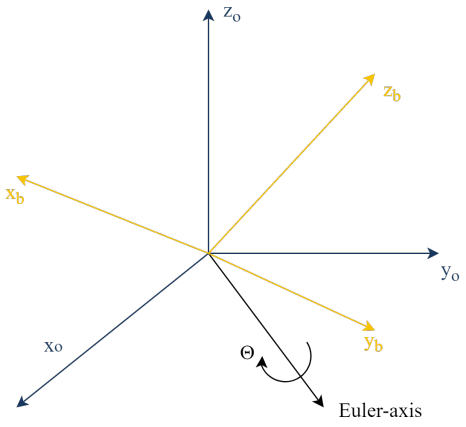


Figure 3.10: Fixed Euler Axis Representation

The quaternion can now be defined as follows:

$$\begin{aligned} q_1 &= e_x \sin\left(\frac{\Theta}{2}\right) \\ q_2 &= e_y \sin\left(\frac{\Theta}{2}\right) \\ q_3 &= e_z \sin\left(\frac{\Theta}{2}\right) \\ q_4 &= \cos\left(\frac{\Theta}{2}\right) \end{aligned} \tag{3.38}$$

with e_x, e_y and e_z the Euler-axis components and Θ the angle of rotation about this axis. From this, it can be seen that the quaternions are not independent, with the following constraint:

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \tag{3.39}$$

The quaternion DCM for ORC to SBC coordinates can be found using this property, and is given as [33]:

$$\mathbf{A}_{O/B} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{3.40}$$

Similar to the Euler angles, the quaternions can be extracted as follows:

$$\begin{aligned} q_1 &= \frac{1}{4q_4}(\mathbf{A}_{23} - \mathbf{A}_{32}) \\ q_2 &= \frac{1}{4q_4}(\mathbf{A}_{31} - \mathbf{A}_{13}) \\ q_3 &= \frac{1}{4q_4}(\mathbf{A}_{12} - \mathbf{A}_{21}) \\ q_4 &= \frac{1}{2}\sqrt{\mathbf{A}_{11} + \mathbf{A}_{22} + \mathbf{A}_{33} + 1} \end{aligned} \tag{3.41}$$

In the case where $q_4 = 0$, the equations need to be recalculated using a different element. The attitude can also easily be propagated using the body angular rate vector relative to the ORC frame,

$$\omega_B^O = \begin{bmatrix} \omega_{xo} \\ \omega_{yo} \\ \omega_{zo} \end{bmatrix} \tag{3.42}$$

and is done as follows:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_{zo} & -\omega_{yo} & \omega_{xo} \\ -\omega_{zo} & 0 & \omega_{xo} & \omega_{yo} \\ \omega_{yo} & -\omega_{xo} & 0 & \omega_{zo} \\ -\omega_{xo} & -\omega_{yo} & -\omega_{zo} & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{3.43}$$

The one negative aspect about quaternions is that it is difficult to be visualised by humans, and are therefore primarily used for attitude propagation where Euler angles

are converted to quaternions beforehand and then back to Euler angles afterwards. This RPY to quaternion transformation matrix is given as:

$$\vec{q} = \begin{bmatrix} C\frac{\psi}{2}S\frac{\theta}{2}C\frac{\varphi}{2} + S\frac{\psi}{2}C\frac{\theta}{2}S\frac{\varphi}{2} \\ S\frac{\psi}{2}C\frac{\theta}{2}C\frac{\varphi}{2} - C\frac{\psi}{2}S\frac{\theta}{2}S\frac{\varphi}{2} \\ C\frac{\psi}{2}C\frac{\theta}{2}S\frac{\varphi}{2} - S\frac{\psi}{2}S\frac{\theta}{2}C\frac{\varphi}{2} \\ C\frac{\psi}{2}C\frac{\theta}{2}C\frac{\varphi}{2} + S\frac{\psi}{2}S\frac{\theta}{2}S\frac{\varphi}{2} \end{bmatrix} \tag{3.44}$$

Once again, C and S are used to represent cos and sin functions, respectively. The basic orbital model has now been discussed; including how satellites are represented in orbit, however, there are external forces which perturb orbits. It was stated earlier that angular momentum of the orbit is constant and thus the orbital plane is fixed in space, but these perturbing forces can change the angular momentum. The three forces with the highest impact on a satellite in LEO, namely atmospheric drag, gravitational forces and solar radiation pressure, will follow.

3.2 Atmospheric Drag

Atmospheric density affects satellites in orbit even without a device whose purpose is to remove the orbital energy of the satellite using the atmosphere. The atmospheric density in LEO is many orders of magnitude smaller than that near the ground, but it is still large enough to impact the satellite. The atmosphere lowers the orbit altitude of satellites with orbit heights of 1000 km or less [33]. The perturbations can have periodic and secular effects on the orbital elements. Secular effects are those that increase linearly with time, while periodic effects increase and decrease intermittently, shorter ones typically repeating with the satellite's orbital period and longer ones lasting a few weeks at a time [10]. Atmospheric drag has significant secular effects on the semi-major axis and the eccentricity. The semi-major axis decreases with time, as such the orbit gets smaller, in turn increasing the atmospheric density around the satellite and increasing the effect of the atmosphere. The change in the semi-major axis per revolution, a_{rev}, for circular orbits is given as:

$$\Delta a_{rev} = -2\pi \left(\frac{C_D A}{m} \right) \rho a^2 \tag{3.45}$$

where C_D is the drag coefficient of the satellite, A is the cross-sectional area affected by the drag, m is the mass of the satellite, ρ is the atmospheric density at the current orbit height, and the last a is the orbit's original semi-major axis. The drag coefficient is typically taken as $C_D = 2.2$ for satellites [10].

Table 3.1: Drag Coefficients for Common Shapes [10]

Shape	Surface	Drag Coefficient
Flat Plate, Face First	Specular	4
	Diffuse	3
Flat Plate, Angle of Incidence α	Specular	$4\cos^2\alpha$
	Diffuse	$2+\cos\alpha$
Sphere	Specular	2
	Diffuse	$8/3 \approx 2.667$

Table 3.1 gives values of the drag coefficient for two common shapes. Specular reflection occurs on a smooth surface while diffuse reflection occurs when the surface has a rough material. The drag emits a force upon the satellite which is necessary to calculate the motion of the satellite. The acceleration of the satellite due to drag, a_D is given as:

$$a_D = -\frac{1}{2}\rho\left(\frac{C_D A}{m}\right)v^2 \tag{3.46}$$

This equation can easily be manipulated into a force that the satellite experiences by multiplying by the satellite's mass on both sides, yielding:

$$F_D = -\frac{1}{2}\rho C_D A v^2 \tag{3.47}$$

The density of the atmosphere changes with the flux density of the Sun. The Sun experiences an 11-year solar cycle in which it cycles between maximum and minimum flux density. The solar cycle can be seen entering a minimum in late 2019 in Figure 3.11.

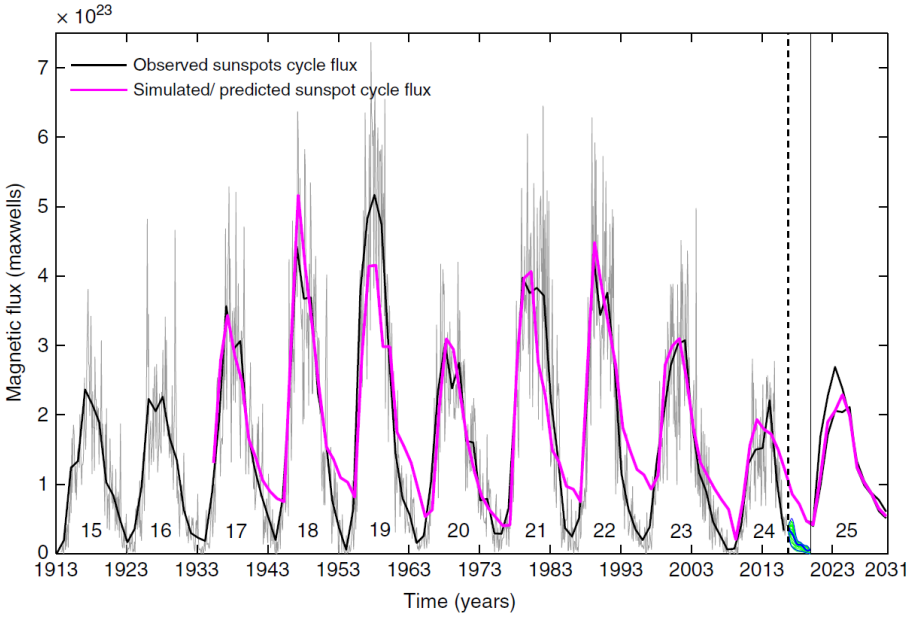


Figure 3.11: Observed and Predicted Solar Flux Density [34]

As the solar flux increases, so does the atmospheric density and vice versa for low solar flux. A satellite will, therefore, take longer to deorbit during a time of solar minimum compared to the time taken in solar maximum, as the drag force will be much smaller during the solar minimum.

3.2.1 Atmospheric Density Models

It is required to model the atmosphere to allow the effects thereof on LEO satellites to be monitored. Three atmospheric density models can be used in STELA, the Semi-analytic Tool for End-of-Life Analysis software, simulations, namely the 1976 US Standard Atmosphere model, the Jacchia Reference 1977 model and the NRLMSISE-00 model. STELA will be discussed further in section 4.2.2. These three atmospheric density models will be discussed and compared to determine which model is best suited for drag analysis.

1976 US Standard Atmosphere Model

The oldest model available in STELA, is the 1976 US Standard Atmosphere model. Crafted by the US Committee on Extension to the Standard Atmosphere (COESA) set up in 1953, prompted 1958, 1962, 1966, and 1976 renditions of the US Standard Atmosphere models. These models were joint publications by NASA, the National Oceanic and Atmospheric Administration, and the US Air Force, but more than 30 American organisations took part in creating the COESA models. The atmospheric densities from sea-level to an altitude of 1000 km are given in this model, from spacecraft data, and based off of ideal gas theory, with a resolution of 5 km and one of 50 m at altitudes below 32 km. The COESA atmospheric models except for the 1966 version, consist of a profile with various parameters that constitute the atmosphere rotating with the Earth as a rigid body for moderate solar activity. The parameters in the profile consist of the density, pressure, temperature and acceleration caused by gravity amongst others. These models can be given in metric or imperial units. [35].

Jacchia Reference 1977 Model

The Jacchia models are another example of the first atmospheric models to be developed. The first model was published as a report by L.G. Jacchia in 1970, after that he updated it in 1971 and 1977. They are also based on rocket and satellite drag data; however, they contain data for altitudes from 90 km up to 2500 km. The reports include tables with parameters for density, temperature and composition. What makes the Jacchia models more unique is that they include auxiliary tables with seasonal, latitudinal, geomagnetic and solar effects, and Jacchia was the first to notice the relationship with solar flux and the atmosphere [36]. The Jacchia model also assumes that the atmosphere is an idealized, steady-state atmosphere; that is that it rotates rigidly with the Earth.

The US Air Force improved the Jacchia Reference 1976 model in 2008, and this model became known as the Jacchia-Bowman 2008 model [32]. This model is, however, not available to use in STELA and was, therefore, not considered in this model.

NRLMSISE-00 Model

Mike Picone, Alan Hedin, and Doug Drob developed the NRLMSISE-00 model, based off of the latest MSIS class of atmospheric density models, MSISE-90 [37]. MSIS stands for mass spectrometer and incoherent scatter radar - the only data sources for development of these models [38], [39]. Hedin developed the MSIS model in the late 1970s, along with the others who started developing models for the first time, with the first model, known as MSIS without a year, published in 1977. After that, the MSIS-83 and MSIS-86 brought improvements to the first model, and the MSIS-86 model replaced Jacchia 71 as the Committee on Space Research's International Reference Atmosphere [39]. All

these models were only for the regions above the thermosphere, which is altitudes of at least 100 km [32]. Afterwards, the model was expanded to the MSISE-90 model, which includes data from space shuttle flights [40] and has parameters down to sea-level [41]. 'E' was added to the name to indicate the model extends from the ground through to the exosphere, which is the outermost layer of the Earth's atmosphere [32]. This model has parameters for densities and temperatures.

The US Naval Research Laboratory (NRL) further improved upon the MSISE-90 model in the year 2000, which brought about the NRLMSISE-00 model. The improvements included, among others, the large-scale use of accelerometer and drag data on total mass density [42]. Like the Jacchia models, these MSIS models use geomagnetic and solar activity to model the temperatures in the exosphere. However, unlike the Jacchia models, the density is not calculated from direct integration with the altitude over the temperature profile. These models use separate independent models for the thermospheric temperature and of each of the atmospheric constituents' (gasses that make up the atmosphere) number densities. These modelled concentrations and thermospheric temperatures are calculated using many terms such as the solar and geomagnetic activity indices, geomagnetic latitude and local geomagnetic time [39]. This improved calculation means that the atmospheric density is much more thoroughly calculated than in the other two older models. Wertz et al. use NRLMSISE-00 in The New SMAD's Earth Satellite Parameters Table [10, pg. 1031].

3.3 Gravity and the Earth's Oblateness

The effect of gravity due to the Earth being an oblate spheroid has the most substantial impact on a satellite's orbit out of the orbit perturbations that will be considered [10]. The Earth's oblateness causes the argument of perigee and the RAAN to drift. It also has a moderate periodic effect on the eccentricity of the orbit. The RAAN drift is sometimes utilised to create Sun-synchronous orbits, which is a nearly polar orbit in which the satellite passes over the same point at the same local time with each pass. Sun-synchronous orbits are very popular for Earth observation missions.

The Earth's radius is 21.4 km larger at the equator than at the poles, which causes a variation in the gravitational forces that a satellite will undergo during its orbit [10]. The mass of the Earth is also asymmetrically distributed, which further causes changes in the gravity that the satellite will experience. The acceleration, and thus the force, of the satellite due to the gravity of a spherically symmetric central body is given as follows [10]:

$$\begin{aligned} \vec{a} &= -\frac{\mu}{r^3}\vec{r} \\ \therefore \vec{F}_G &= -\frac{\mu m}{r^3}\vec{r} \end{aligned} \tag{3.48}$$

The oblateness can be factored in using what is known as the J_2 perturbation, where "J_2" and "Earth Oblateness" can be used interchangeably. It is the third coefficient in the geopotential model of the Earth, which is a set of coefficients in the set of the spherical harmonic expansion. Essentially, the geopotential model splits the Earth into different sections or zones based on the coefficient, with each zone having the opposite sign to the zones next to it. The first coefficient, J_0, represents a point mass, while J_1 has a sign-change over the equator, thus representing the mass difference in the northern and southern hemispheres. J_2 has two sign-changes between north and south and as such

represents the equatorial bulge's mass distribution [10]. The acceleration due to the J_2 perturbation in the inertial frame, with the subscripts i omitted due to spacing, is given as follows [31]:

$$\vec{a}_{J_2} = \frac{3J_2\mu R_E^2}{2r^5}\begin{bmatrix}\left(-1+\frac{5z^2}{r^2}\right)x\\ \left(-1+\frac{5z^2}{r^2}\right)y\\ -3\frac{5z^2}{r^2}\end{bmatrix} \tag{3.49}$$

with the J_2 term defined in equation 3.50, μ is the Earth's gravitational constant, R_E is the equatorial radius of the Earth, r is the magnitude of the satellite's position vector in ECI, and x, y and z are the position vector's elements in ECI. The J_2 effect is the largest one as its coefficient is many orders of magnitude larger than the J_3 and J_4 terms. The values of the first five geopotential model terms are given in equation 3.50:

$$\begin{aligned} J_0 &= 1\\ J_1 &= 0\\ J_2 &\approx 1.083\times10^{-3}\\ J_3 &= -2.54\times10^{-6}\\ J_4 &= -1.61\times10^{-6}\end{aligned} \tag{3.50}$$

It is, however, more accurate to include the J_3 and J_4 effects as they cover the other mass distributions that are not equal besides just the equatorial bulge. The accelerations due to these two effects in ECI are given as [31]:

$$\vec{a}_{J_2} = \frac{5J_3\mu R_E^3}{2r^7}\begin{bmatrix}\left(-3z+\frac{7z^3}{r^2}\right)x\\ \left(-3z+\frac{7z^3}{r^2}\right)y\\ 6z^2-\frac{7z^4}{r^2}-\frac{3r^2}{5}\end{bmatrix} \tag{3.51}$$

$$\vec{a}_{J_2} = \frac{15J_4\mu R_E^4}{8r^7}\begin{bmatrix}\left(1-\frac{14z^2}{r^2}+\frac{21z^4}{r^4}\right)x\\ \left(1-\frac{14z^2}{r^2}+\frac{21z^4}{r^4}\right)y\\ 5-\frac{70z^2}{3r^2}+\frac{21z^4}{r^4}\end{bmatrix} \tag{3.52}$$

These spherical harmonics for J_1 to J_4 are visually depicted on Jupiter in Figure 3.12. The figure illustrates where the mass distribution is made to be more and less on the planet, using positive and negative signs for positive and negative mass distributions, respectively. As mentioned earlier, J_1 represents the standard spherical model with the sign change over the equator. J_2 represents less mass around the poles (negative signs) and more around the equator (a positive sign), resulting in an oblate spheroid or an equatorial bulge. The J_3 spherical harmonic expansion has four sections across the sphere, with a positive sign at the northern pole and alternating signs down to the southern pole. This expansion results in an egg-shaped sphere. The J_4 expansion is similar to J_2 in that it has a positive mass distribution over the equator with negative ones above and below it. However, it also has positive signs over both poles, resulting in the diamond-shaped sphere in the figure.

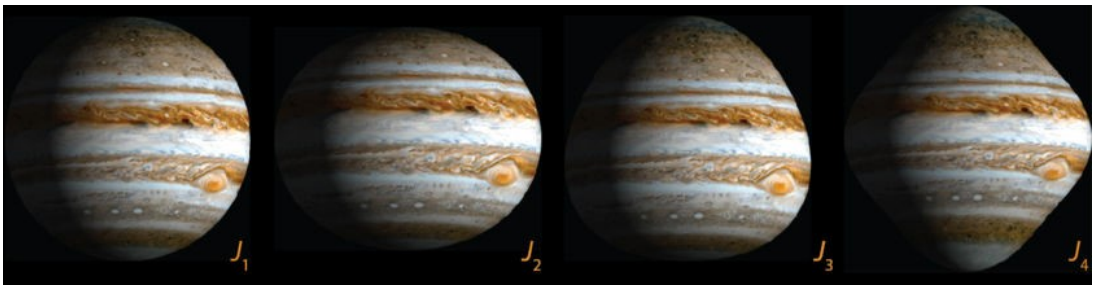


Figure 3.12: Spherical Harmonics Depicted on Jupiter [43]

These accelerations can all be combined by adding them together to form $\vec{a}_{OB}$. This acceleration can then be multiplied by the satellite's mass to form the force which acts upon the satellite due to gravity and the Earth's oblateness, $\vec{F}_{OB}$.

3.4 Solar Radiation Pressure

The final external perturbation affecting a satellite's orbit to be considered is the perturbation effect due to SRP from the Sun. Photons from the Sun travel across the solar system as solar wind, and when they hit the satellite, it results in a force that pushes the satellite in the direction away from the Sun. Due to this force being rather small, it is usually only of importance for small, light satellites with large areas exposed to the Sun. That is the case for a CubeSat with a deployed deorbiting sail or balloon. However, when the satellite does not experience a large drag force, at high altitudes the SRP force can aid in its deorbiting. The solar wind can interact with the surface of the deorbiting device in four different ways [32]:

1. The surface completely absorbs the radiation, called pure absorption.
2. The radiation undergoes specular reflection.
3. The radiation undergoes diffuse reflection.
4. The surface allows the radiation to pass through it completely, called transmission.

The electromagnetic radiation pressure's magnitude is dependent on the ratio of Sun's power density per square metre at 1 AU - which is the mean distance between the Earth and the Sun - to the speed of light in a vacuum. In other words:

$$p = \frac{\Phi}{c} \tag{3.53}$$

At 1 AU, the Sun's power density is $\Phi \approx 1371 W/m^2$, and thus the SRP magnitude at the Earth is $p = 4.6 \times 10^{-6} Pa$. The incidence angle of the solar wind is given as α. The cosine of the incidence angle can be defined as the dot product of the Sun vector in SBC and the deorbit device's normal angle, namely:

$$\cos\alpha = \vec{S}_b \cdot \vec{n}_{device} \tag{3.54}$$

The sine of the incidence angle is given as:

$$\sin\alpha = \sqrt{1 - \cos^2\alpha} \tag{3.55}$$

For specular reflection, the total force $\vec{F}_{SRP}$ is the sum of the incidence and reflected components, $\vec{F}_{SRPi}$ and $\vec{F}_{SRPr}$, respectively [32]:

$$\begin{aligned}\vec{F}_{SRPi} &= p\cos\alpha A(-\sin\alpha\vec{t} - \cos\alpha\vec{n}) \\ \vec{F}_{SRPr} &= p\cos\alpha A(\sin\alpha\vec{t} - \cos\alpha\vec{n}) \\ \vec{F}_{SRP} &= -2p\cos^2\alpha A\vec{n}\end{aligned} \tag{3.56}$$

where A is the cross-sectional area that the solar wind reacts with, and $\vec{t}$ and $\vec{n}$ are the device's tangential and normal directions respectively. The chosen material, the material used for NASA's Jet Propulsion Lab (JPL) solar sail, does not have an entirely specular reflection [44]. The normal and tangential forces will be similar to that of a specular reflection, but the various coefficients will cause a slightly different force to act upon the device. Using the JPL sail material for the device yields the following normal and tangential SRP forces [44], [45]:

$$\begin{aligned}\vec{F}_{SRPn} &\approx 1.83pA\cos^2\alpha\vec{n} \\ \vec{F}_{SRPt} &\approx 0.17pA\cos\alpha\sin\alpha\vec{t}\end{aligned} \tag{3.57}$$

Chapter 4

Hardware Design and Simulation Environments

This chapter covers the description of the hardware used for physical device testing and the simulation environments used for the deorbiting analysis of the chosen device. First, all the different components of the hardware will be considered and the role they each play in the design. Then the environments used for simulating and analysing the deorbiting process will be discussed.

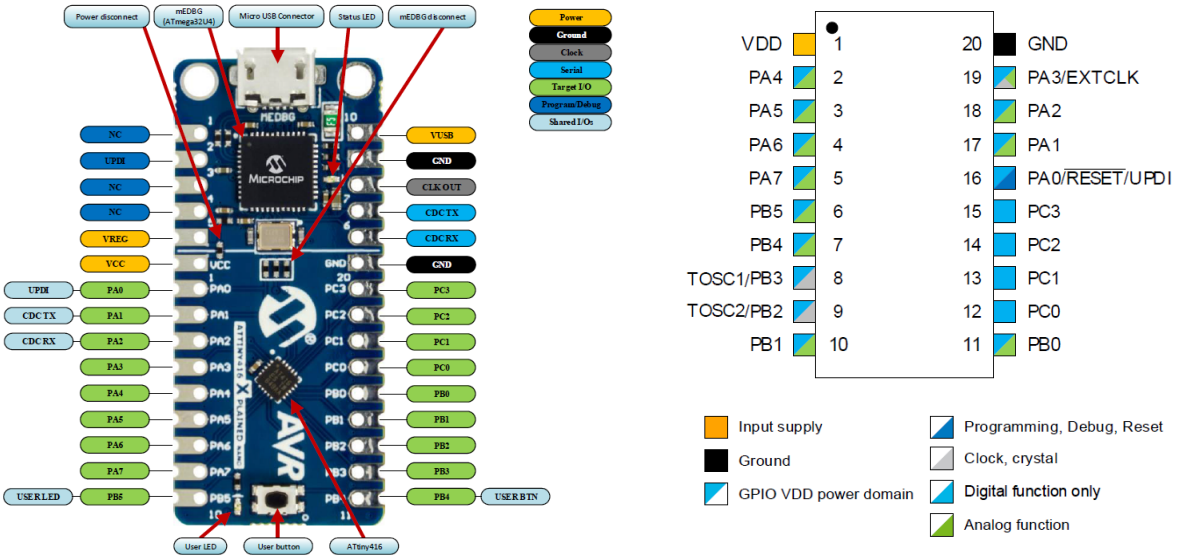
4.1 Hardware Description

The deorbiting device chosen for this study was one that uses atmospheric drag as a deorbiting method, namely a balloon. The balloon was chosen as the deorbiting device above a sail due to it being more straightforward to deploy than a sail, which required booms to unfold it, compared to gas let into the balloon for inflation. The balloon cannot be made from plastic as it needs to be able to survive in the vacuum of space. A store-bought Mylar balloon was chosen for the prototype device. The volume of this balloon was around 6.6 ℓ; however, just 100 ml of Nitrogen gas was needed to fill up the balloon in vacuum-like conditions of 1.5 mbar.

The main requirement of the hardware was that it needed to be low-powered, as the device would be powered from batteries separate from the main power supply, for many months or a few years before being activated. The controller would stay in sleep mode until it is required for deployment of the balloon. The deployment can be triggered either by having the main satellite send a message once a month, essentially "checking in" with the device, and when three messages are missed in a row the balloon is deployed. Another way would be setting a timer for the mission duration so that the balloon deploys thereafter, but this would mean it would deploy even if the satellite was still able to perform its mission longer than the original duration. Each version of the hardware was done on a copper printed circuit board (PCB). The schematics and the layouts for these PCBs can be found in Appendix B. The description of the microcontroller, valve and various other hardware components and their purposes follow below.

Microcontroller

The ATtiny416 Xplained Nano evaluation board was used to program Microchip's ATtiny416 microcontroller. The evaluation kit allows the ATtiny416 to be programmed



(a) ATtiny 416 Xplained Nano Layout [46] (b) ATtiny 416 Xplained Nano Pinout [47]

Figure 4.1: ATtiny416 Xplained Nano Overview

directly with an on-board embedded programmer and thus can be done without any external tool. It is a 20 pin small-outline integrated circuit (SOIC) with multiple analogue and digital pins that can be utilised for the device. The operating voltage, V_{CC}, can be between 1.8 V and 5 V and was chosen at 3.3 V. The layout and pinout are shown in Figure 4.1 (a) and (b).

Various peripherals were utilised for the device. The clock and timer peripherals were used for timing the opening and the closing of the valve, and how long it should stay that way. An interrupt service routine was implemented. The timer used is the 16-bit timer, one of three available timers on the microcontroller. The analogue-to-digital converter (ADC) was used to read in values coming into the microcontroller from the instrumentation amplifier. The reference voltage of the ADC was set to 1.5 V. Lastly, the universal synchronous asynchronous receiver-transmitter (USART) was used for communications between the laptop or computer and the device during tests. The baud rate was set to 19200 baud, because of the long cables that are in the vacuum chamber. Higher baud rates may have resulted in errors on the receiving end of the transmission due to the capacitance in the cables affecting the data. The USART received the instrumentation amplifier values from the microcontroller and sent back commands to open or close the valve.

The ATtiny416 chip itself is an 8-bit AVR processor which can run up to 20 MHz. There is an ultra low power sleep mode where the processor runs at 36 kHz. This standby sleep mode can turn off any peripherals and turn them back on when woken up again. The peripherals that are not disabled, such as the USART, and the watchdog timer can wake the device from sleep mode using interrupts. This functionality is beneficial during the satellite's mission lifetime when the deorbit device will not be active.

Valve

The valve to control the airflow to the balloon was chosen as a latching solenoid valve from The Lee Company, more specifically their LHLA0342311H valve. It is a two-port, normally closed, plug-in valve with an actuating voltage of 3.3 V. A latching valve keeps

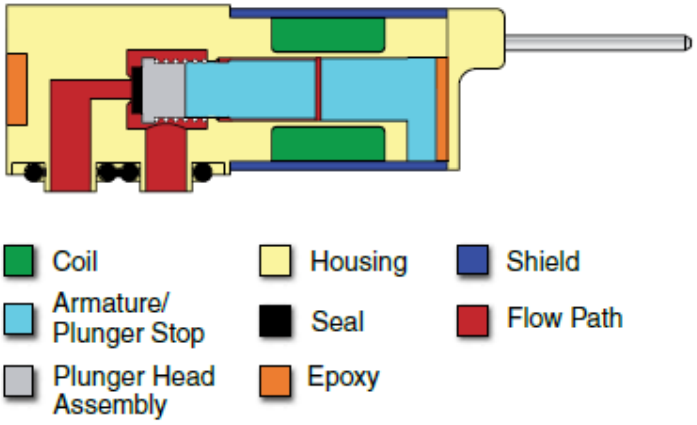


Figure 4.2: Two-Port Latching Valve Model [48]

its position, be it open or closed, with a voltage pulse. This latching means that constant power is not required over the valve when it is in the position that the current generates. A latching valve was chosen to reduce power consumption further. This valve starts in closed position and only requires a pulse of 10 ms to 30 ms in duration to open it. The shortest time of 10 ms was chosen. A pulse with the opposite polarity will then switch the position back to the closed one [48]. Figure 4.2 shows the model for The Lee Company's two-port latching valves.

It was decided that the voltage cannot drop by more than 1 V during the pulse. This change in voltage was used to calculate the capacitor needed over the load supply voltage, V_{BB}. The current would not exceed 300 mA as the maximum current supplied by the microcontroller is 200 mA and the extra 100 mA was added for safety. The capacitance was calculated using the formula for the current flowing through a capacitor:

$$\begin{aligned} I &= C\frac{\Delta V}{\Delta t} \\ \therefore C &= I\frac{\Delta t}{\Delta V} \\ &= 0.3\frac{0.01}{1} \\ &= 3000\mu F \end{aligned} \tag{4.1}$$

A 0.01 μF capacitor was also added in parallel to the 3000 μF one to handle noise and the fast transients of the pulse. The minimum amount of time between opening and closing the valve is equal to one second. The resistor value was chosen using $T = RC$ and thus equals 330 Ω. Figure 4.3 shows the simple RC circuit design used for the calculations.

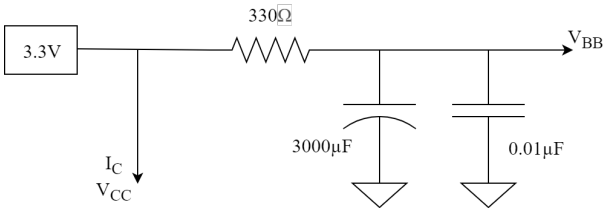


Figure 4.3: Resistive Capacitive Circuit Schematic

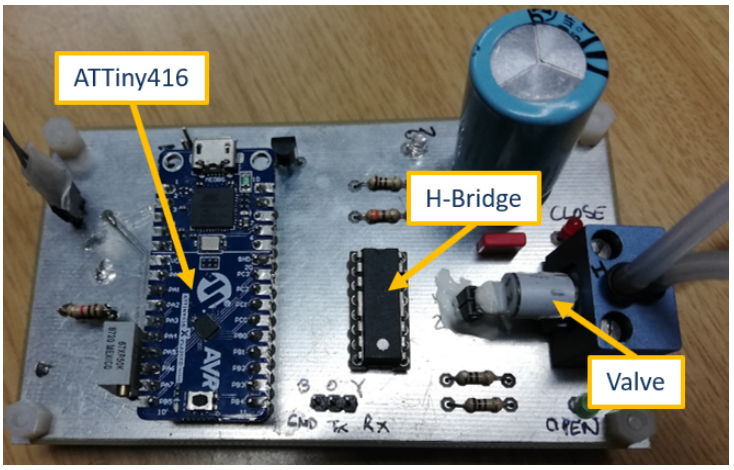


Figure 4.4: Top of the Hardware Design PCB Version 5

Figure 4.4 is a photo of the top of the final version of the PCB with all the components on it. The top contains the microcontroller, the H-bridge, the valve, a 3.3 V voltage regulator, the capacitors and resistors, LEDs to indicate operation and which state the valve is in, as well as the connection pin headers for the communication. The three components discussed above are highlighted in the figure.

H-Bridge

The H-bridge is used to control the opening and the closing of the valve. The Allegro 16-pin dual in-line package (DIP) A4973 full-bridge PWM motor driver chip was used in this project. The logic supply voltage, V_{CC}, is rated for voltages between 3 V and 5 V and the load supply voltage is rated up to 50 V [49]. It is used to send voltages with changing polarities to the pins of the valve, enabling the valve to change its state. The voltage supply to the load comes from V_{BB} shown in Figure 4.3, and thus it was chosen to operate at 3.3 V, the valve's actuating voltage. V_{CC} was chosen to have the same voltage for simplicity.

Strain Gauge

A strain gauge is a sensor whose resistance changes according to how much it bends. The strain gauge in this study has a nominal resistance of 120 Ω. The strain gauge was used to measure the amount the balloon has filled up with gas, based off the curvature of the balloon's outer surface. At first, the strain gauge was placed flat and measured the curve of the balloon as it fills, but there was not a significant difference noted. Thereafter, it was placed on a curved piece of metal which flattened as the balloon inflated, which yielded a more substantial difference in resistance. The difference was measured using a Wheatstone bridge, which is a circuit of four resistors with one unknown, used to measure the unknown one. The layout of the Wheatstone bridge is shown in Figure 4.5.

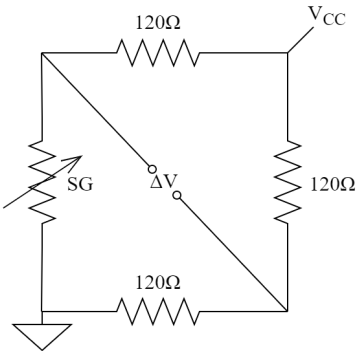


Figure 4.5: Wheatstone Bridge Circuit Schematic

Instrumentation Amplifier

An instrumentation amplifier was used to amplify the signals coming in from the Wheatstone bridge. The instrumentation amplifier chosen for this study was the eight-pin very-thin shrink small-outline package (VSSOP) Texas Instruments' INA333 instrumentation amplifier. It is a low power amplifier, with a quiescent current of 50 μA [50]. The two inputs are connected above the two lower sides of the Wheatstone bridge, depicted in Figure 4.5 as ΔV. The instrumentation amplifier sends the voltage to the microcontroller's ADC pin to be compared.

The schematics of all the versions of the boards are in Appendix B.1 as mentioned previously, and Figure 4.6 shows the board layout for the final version of the design done in EAGLE, which is software for creating PCB designs. This board layout can also be found in Appendix B.2. The placements for the strain gauge, Wheatstone bridge and instrumentation amplifier are annotated in the figure. Blue lines indicate connections on the bottom layer of the double-sided PCB while the red lines are those on top of the PCB.

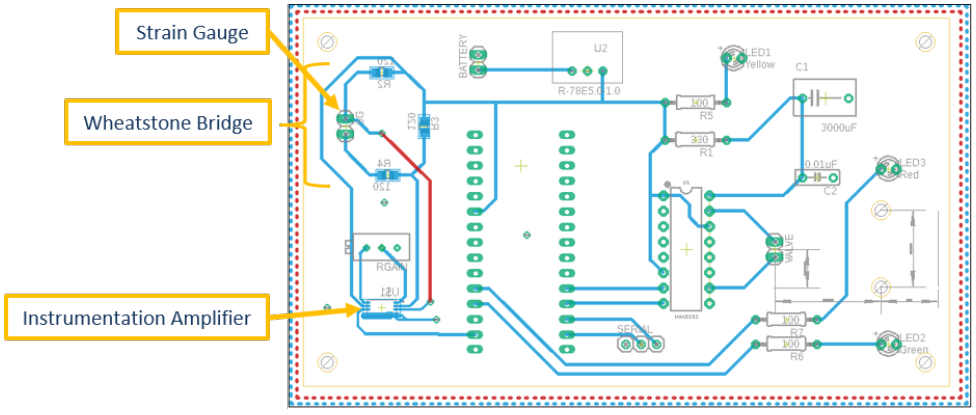


Figure 4.6: PCB Version 5 Layout in EAGLE

The hardware tests were performed in a small vacuum chamber where the strain gauge voltages were recorded on a laptop that was communicating with the microcontroller. The pressure values inside the chamber had to be read manually as the chamber itself has no interface to connect to a computer, and because the pressure is given on an electronic display without an external interface. The pressure values were recorded every 30 seconds, starting from low values such as 200 mbar until it reached 1 mbar. The pump to create the vacuum would then be shut off, and the pressure would stabilise around 2 to 10 mbar. The

balloon was inflated during various tests for a few seconds at a time, and in final tests the inflated balloon - not in full atmospheric pressure, but inflated in vacuum conditions - was left inside the chamber for long durations. During one test, the temperature inside the vacuum chamber was also recorded. There were also three tests conducted in atmospheric conditions done in the lab to compare the balloon and strain gauge's behaviour to that from the vacuum chamber.

4.2 Simulation Environment

It is entirely infeasible to physically test multiple balloons on multiple satellites with various weights and orbit heights. It is, therefore, necessary to implement a simulation environment in which deorbit times can be predicted rather than measured. A MATLAB and Simulink program (hereafter referred to as the MATLAB simulation or program) was implemented for use with a balloon as the passive deorbiting device. The results will be compared against those with dynamic yawing motion, a sail as well as those from STELA. The two programs will be discussed in this section.

4.2.1 MATLAB

Prof. WH Steyn provided a solar sailing MATLAB script and Simulink program. This program was used to test a solar sail with JPL material specifications on the POSAT satellite. The initialisation (init) code and the Simulink blocks for this program can be found in section C.1 and D.1 of Appendix C and D, respectively. This sail was edge-on to the movement direction of the satellite and thus did not experience much drag. It also used a constant atmospheric density for the specific altitude of the satellite.

This code was modified for the study to serve as a deorbit simulator for a balloon as the device; to determine the balloon size needed for the satellite's size and weight. The atmospheric densities during mean solar activity were added to the code from [10]. As such, this simulation can work for satellites of up to a 1000 km orbit altitude. These densities are given on average at 25 km resolution at lower altitudes and a 50 km resolution for higher altitudes. These values were interpolated for a finer resolution of 5 km. The oblate spheroid shaped balloon's size is defined with three radii, a, b, and c; and the average weight of the device - along with the hardware - was selected at 500 g. When two circular pieces of material are connected and filled with air, it forms an oblate spheroid with the minor axis around half of the major axes. As such, the balloon's larger two radii were chosen to be the same, and the drag would act upon this area. The optical characteristics of the balloon were chosen to stay the same as the original sail's material, that is the JPL sail material.

A two-line element set (TLE) is a format for encoding the orbital elements of a satellite at a specific point in time known as the epoch. Using an algorithm and suitable perturbation model, the satellite's position and velocity can be calculated. The epoch of this study was chosen as the 29th of August 2019, which is related to solar activity to be explained later on. Code provided by Prof. WH Steyn was used to get the velocity and position vectors of satellites at three different orbit heights, roughly 480 km, 600 to 800 km (elliptical) and 950 km. The 480 km orbit was chosen as the control orbit to be compared to all the other simulations. This code can be found in appendix section C.4. These vectors were used to get the norms, magnitudes and altitude of the orbit. Figure 4.7 shows the breakdown of a TLE.

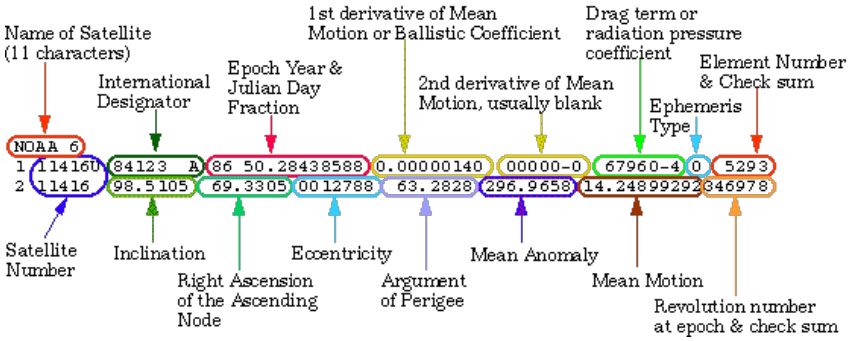


Figure 4.7: TLE Definition [51]

The "Drag" block in the original Simulink model was changed to calculate the drag force while the satellite was deorbiting, thus having the drag increase as it came closer to the Earth. The values of the atmospheric density were those from the mean solar activity in the init file. The "SRP" block in the Simulink model was modified to include the β angle, the angle between the Sun's position vector and the satellite's position vector. The program was also modified to include angular rates of the satellite, adding a block to calculate the attitude of the satellite at each time step. This block takes the RPY angles and body rates in rad and converts them to quaternions for more straightforward update calculations. The quaternions are then converted to a DCM for the drag and SRP calculations. The body rates of the satellite can be constant - although unlikely - or dynamic, which results in a more realistic simulation.

The mean solar activity atmospheric density values are valid for long-duration deorbits. As such, it was decided to include minimum solar activity density values for the shorter deorbit times and to determine the worst-case scenario for deorbiting. Later, it was decided to utilise a dynamic solar activity model with the atmospheric density increasing and decreasing with the solar cycle. A separate Simulink program was made and is referred to as the dynamic case. The program using only the mean and minimum values is referred to as the static case. The init file for both of these cases is the same and can be found in Appendix C section C.2. The code in the "Drag" block was modified to adjust the atmospheric density at the specific height at the specific time. The epoch was selected as the 29th of August 2019 due to a solar minimum starting at around this time, which, once again, gives the worst-case scenario for deorbiting a satellite. The dynamic body rates were also only implemented in the dynamic case as this became the control case to be compared to all the other results. The Simulink blocks for the static and dynamic case can be found in Appendix D section D.2.

A second version of the MATLAB program was made to use a deorbit sail instead of a balloon. The dynamic case's "Drag" and "SRP" blocks were modified for the new area calculations. The sail simulations were not going to include angular rates as it was just used to compare against balloons of the same size that are not spinning or tumbling. As such, the dynamic body rates were not implemented in the Simulink program, saving computation time, as more blocks slow down the simulation.

Three CubeSats sizes were simulated with four different balloon sizes, at one of the three chosen orbit heights. The CubeSats were chosen as 3U, 6U and 12U in size with each having it's maximum allowable mass. The largest balloon had x, y and z radii of 1 m, 2 m and 2 m respectively. These were halved for the second largest size and halved again for the next smaller size. The smallest radii were chosen as one-tenth of the largest ones, resulting in radii of lengths 0.1 m, 0.2 m and 0.2 m, which is similar to the result of

halving the radii of the second smallest balloon.

The 480 km orbit was also simulated with the three CubeSat sizes in dynamic solar activity conditions with a changing yaw angle value. The yaw rate was changing with a sine wave and had three maximum values it could reach, namely 1 °/sec, 5 °/sec and 10 °/sec. These values were chosen to represent realistic yaw rates for a CubeSat with a back-and-forth tumbling motion, as the balloon would prevent excessive motions of more than 10 °/sec. These simulations were compared to those of the same satellites without any change in attitude.

The sail was simulated with lengths and heights equal to double the y and z radii of the balloon. A second sail simulation with lengths and heights that resulted in areas nearly equal to the balloons cross-sectional areas that would impact with the atmosphere was also done. This second simulation would give more comparable results to the balloon compared to the first one, which has sails with slightly larger surface areas. The sail was only implemented in the 480 km orbit with dynamic solar conditions.

4.2.2 STELA

STELA, the Semi-analytic Tool for End-of-Life Analysis software, is a program developed in Java and designed by the French Space Agency for analysing the EoL of a satellite. It enables long-term propagation of satellites in LEO, GEO or a geosynchronous transfer orbit based on semi-analytical models and the UN criteria for deorbiting a satellite within 25 years of EoL. STELA can only implement a sail as a deorbiting device. The program outputs a file with a summary of the computation, including the initial and final orbits and satellite characteristics. It can also convert a TLE file into the satellite parameters, so all the information does not have to be entered manually [52]. As mentioned before, three atmospheric models can be used in the program. Due to the NRLMSISE-00 model being implemented in the MATLAB simulation, it was decided to use this model for the STELA simulations as well. It was decided that the MATLAB simulations had to be compared to simulations from an existing software to determine if the results were realistic or not. STELA is a free software that was easy to implement in this project.

Chapter 5

Results Discussion

The physical and simulated methods of testing have been discussed, and the results will follow. The discussion will start with the hardware tests done in the vacuum chamber as well as results from ambient tests in the lab or atmosphere-level tests. Thereafter, the results from the MATLAB simulations versus those from STELA will be analysed.

5.1 Physical Results

Thirteen tests were conducted with the hardware, three in the atmosphere and the rest in a vacuum chamber. The first vacuum chamber experiment tested the system with the strain gauge glued directly onto the balloon. This test was used to test the valve system in vacuum-like conditions. The balloon was inflated using nitrogen gas. It was noted that the strain gauge did not give a large change in measured output when the balloon inflated. The second test was done with the strain gauge glued onto a piece of flat plastic sheet which was in turn glued onto the side of the balloon that would produce the biggest deformation of the strain gauge when inflated. The balloon still had the nitrogen gas in it from the first test. However, the vacuum tank's heater element was accidentally turned on and the Lithium-ion battery was heated, causing an unreliable test, as the battery expanded and lost its charge very quickly. It was also noted that the plastic did not bend enough for a noticeable change in the strain gauge voltage to be measured. The second test's results are in Figure 5.1.

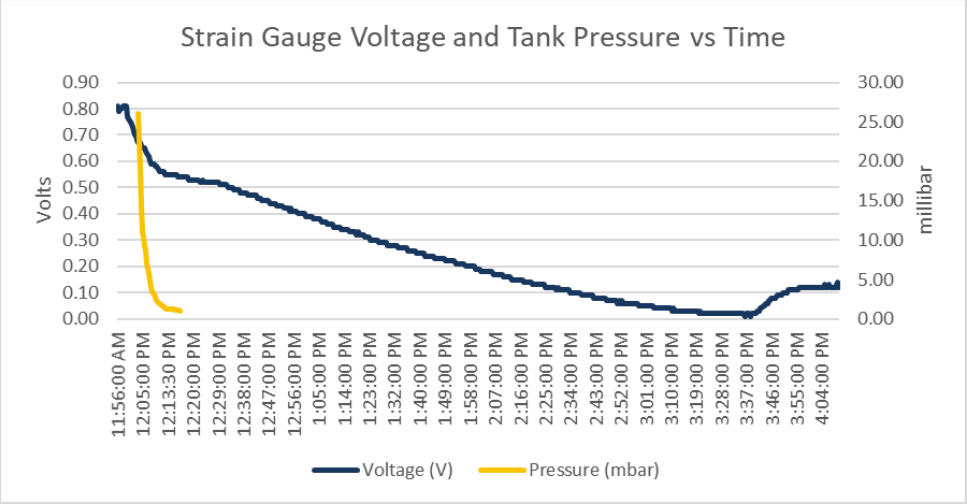


Figure 5.1: Test 2: Vacuum Chamber Results

The navy blue line shows the voltage read in from the ADC coming from the strain gauge and the golden line represents the pressure in the vacuum tank. This colour scheme is used for all the graphs in this section. The pump was also switched off at 1 mbar pressure in the tank. This pressure value is the pump switch-off point for all the tests.

The piece of plastic was curved for the third test. It was placed such that the curve went with the curve of the balloon. The plastic would then straighten out as the balloon inflated. This placement was, however, unsuccessful as the strain gauge did not straighten out enough. For the fourth test, the curved plastic was placed so that the curve was opposite to the balloon's curve. This placement would ensure a broad range of movement for the strain gauge and showed a better result with the change of strain gauge voltage as the balloon inflated as the pressure decreased. This balloon was still the same and thus had enough gas inside the balloon for it to inflate at low pressures. This test's strain gauge voltage and vacuum tank pressure over time are shown in Figure 5.2.

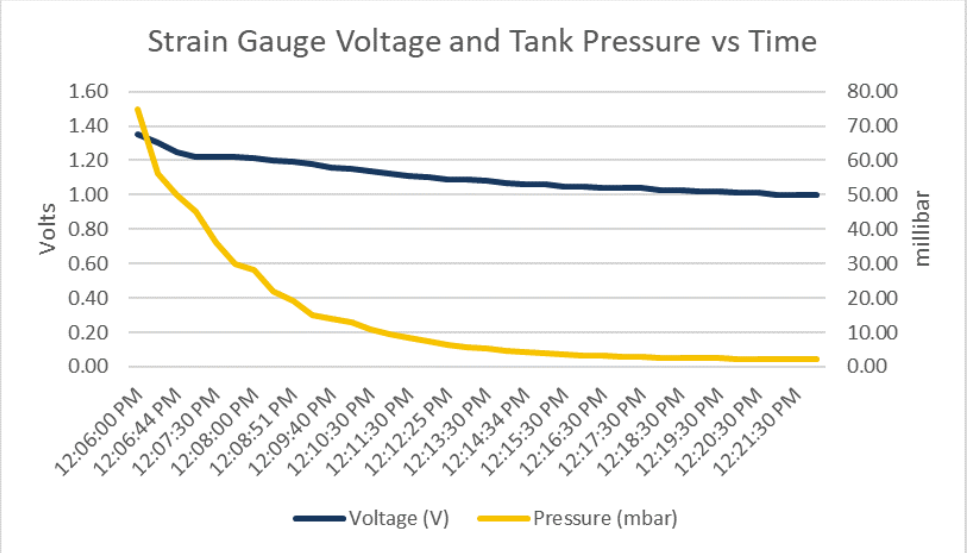


Figure 5.2: Test 4: Vacuum Chamber Results

The fifth test had a new balloon and new strain gauge. The strain gauge was placed on a curved piece of metal and curve was placed with the curve of the balloon, over a part that would straighten out when inflated. Test 5 looked promising until the strain gauge came loose from the balloon and the test was stopped. The results until this point are given in Figure 5.3. The voltage spike at the end is the point where the strain gauge came loose.

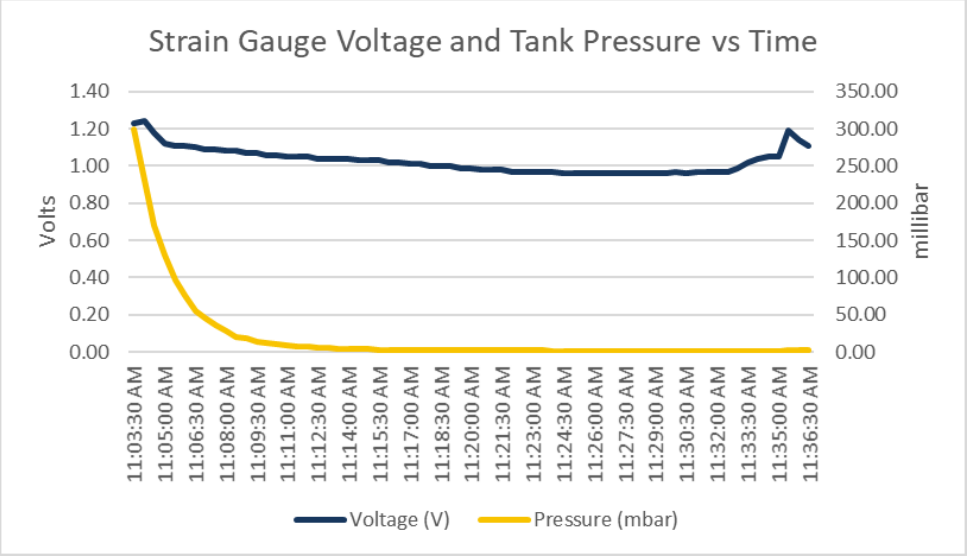


Figure 5.3: Test 5: Vacuum Chamber Results

The strain gauge was secured to the balloon on the curved piece of metal once again, but the sixth test was also unsuccessful as the pins connecting the strain gauge to the PCB came loose when the balloon inflated entirely. This disconnection resulted in the ADC reading in its reference voltage of 1.5 V. Once the pins were secured, the seventh test could be done. The results of the seventh test, a longer test, are displayed in Figure 5.4. There are four data markers on the voltage line. The first marker shows where the balloon started inflating due to previous air still being inside it. The second marker is the point just before two pulses of air got sent into the balloon. The pulses each had the valve open for two seconds. The third marker shows the point where the two pulses had finished. The final marker shows the point where the pump was switched off in the tank to stop the depressurisation. The pressure line now has point markers added to it because the pressure value was recorded at random points during the vacuum state. The tank was re-pressurised after the final pressure reading of 4.7 mbar at 14:27:00.

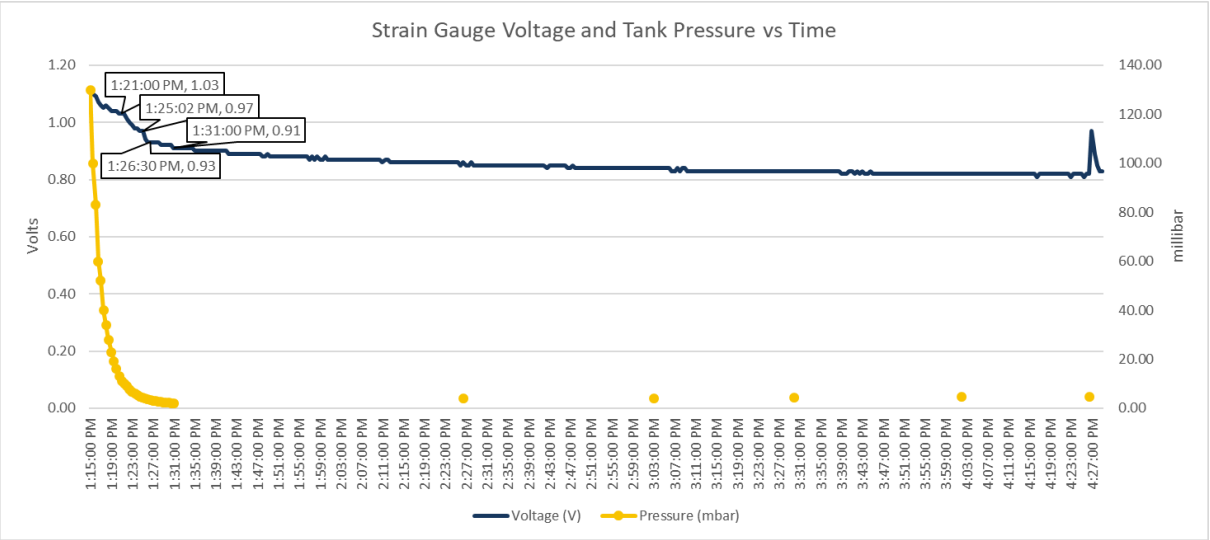


Figure 5.4: Test 7: Vacuum Chamber Results

The voltage spike at the end was, once again, caused by the vacuum tank re-pressurising. This cause of the voltage spike is the case for all further vacuum chamber tests and will not explicitly be mentioned again. It can be seen from test 7's results that the strain gauge's voltage continued to drop even though no more air was pumped into the balloon and the pressure no longer decreased. The voltage after the pump has been turned off can be seen more closely in Figure 5.5.

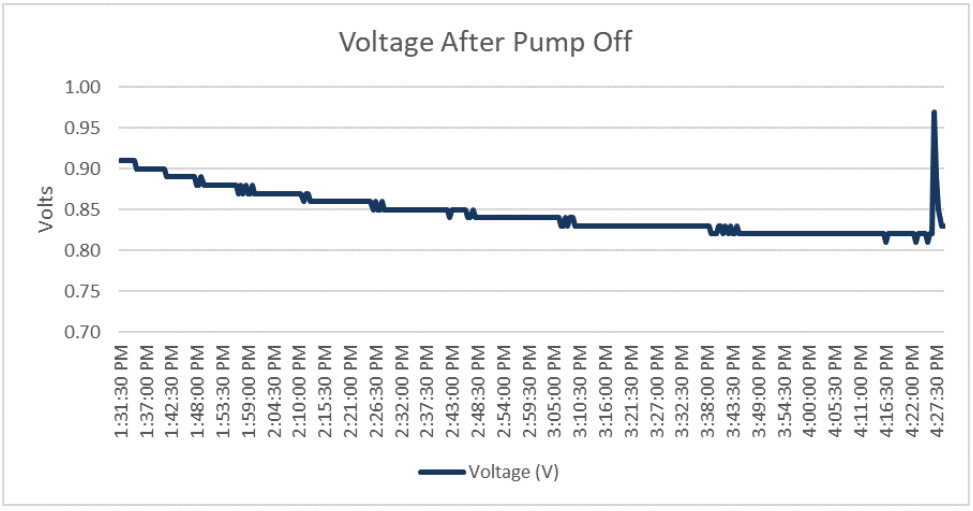


Figure 5.5: Test 7: Strain Gauge Voltage when Holding Vacuum

It was decided to test the temperature inside the vacuum chamber to figure out why this voltage decrease was happening, in case the temperature had slowly been increasing, similarly to when the heater had been on in test 2. A temperature sensor was placed inside the vacuum chamber for test 8, which took the temperature reading every ten minutes. The temperature is shown as red circles on the graph in Figure 5.6. The strain gauge voltages were lost for the depressurising portion of the test, as such the depressurisation voltages from tests 5 to 7 were averaged and used for that section of test 8. These voltages are shown in a lighter shade of blue.

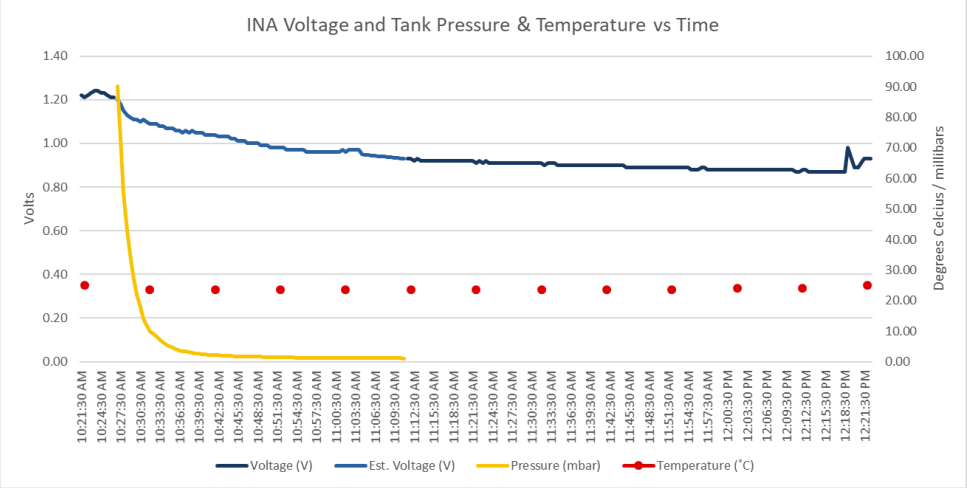


Figure 5.6: Test 8: Vacuum Experiment with Temperature Measurements

It can be seen from the graph that the temperature stayed constant at 23.61 °C during the test. As such, this was not the cause of the decreasing voltages. Another two-hour test was conducted to see if the results were any different, but they were very similar to test 7. No more air was added to the balloon during test 9. The results of this test can be found in Figure 5.7.

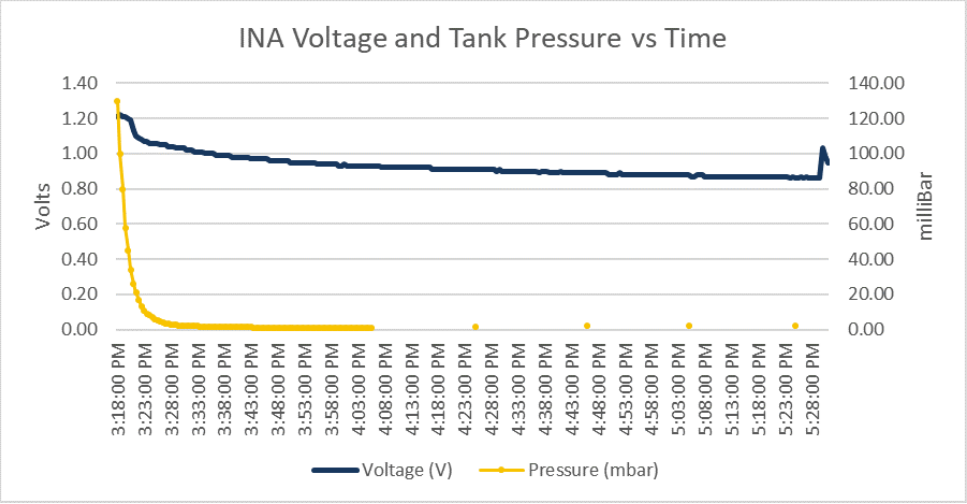


Figure 5.7: Test 9: Vacuum Chamber Results

The next three tests were done in the lab at normal atmospheric levels. The first lab test lasted a few hours but was interrupted due to the connection accidentally being broken. A second lab test was conducted overnight. The results of this test are in Figure 5.8.

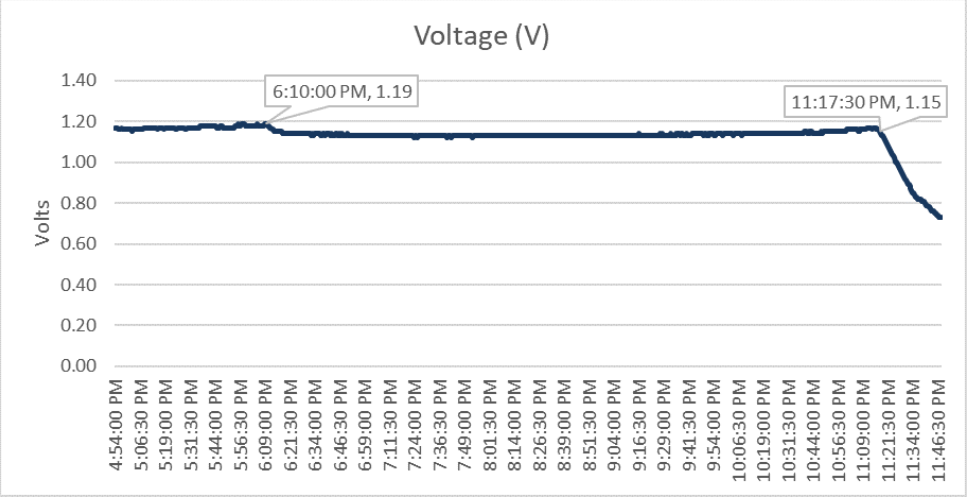


Figure 5.8: Second Lab Test Results

It seems that the system is slightly sensitive to movement as the voltages before the first data marker at time 18:10:00 are noisy compared to those after this marker once the lab was empty. The voltage stayed constant at around 1.13 V during the night and did not decrease until the battery started to run flat around 23:17:30. A second overnight test was done to confirm these results. Figure 5.9 shows the results from this third lab test, and a similar pattern can be seen compared to the first overnight test. The only difference is that the voltages were slightly lower than the first test's voltages. The system settled at around 1.08 V for the night run until the battery became flat once again.

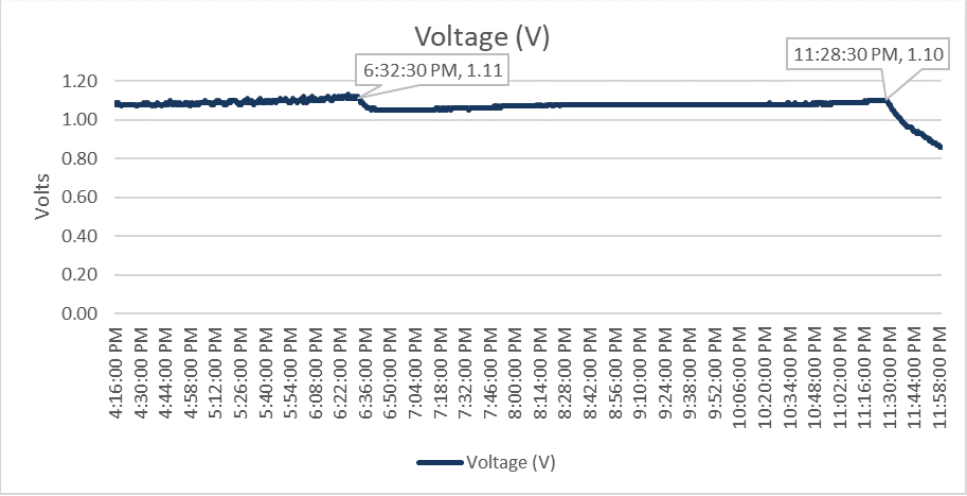


Figure 5.9: Third Lab Test Results

A final vacuum chamber test was conducted after this. This test was an extended duration test to determine the value that the strain gauge voltage settles on. The pump had to be switched off just before the tank reached 1 mbar of pressure due to loadshedding being implemented. The final test results are depicted in Figure 5.10.

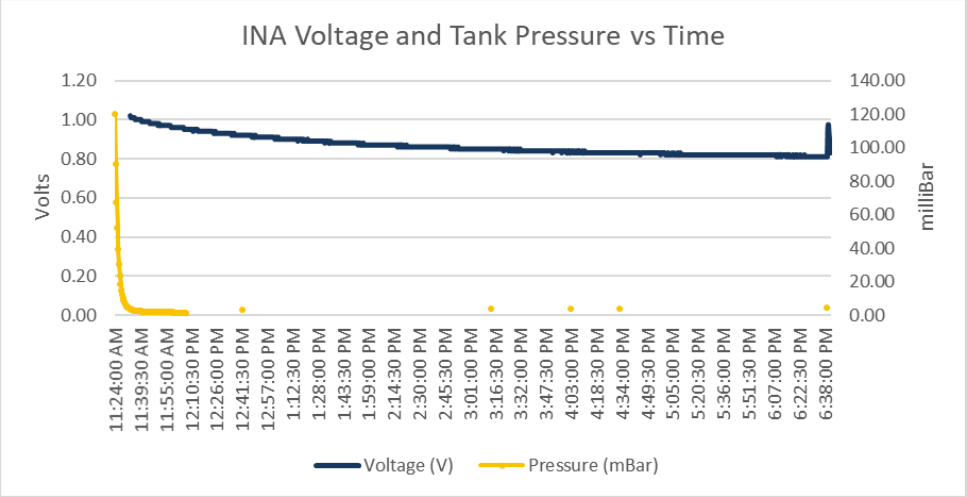


Figure 5.10: Test 10: Vacuum Chamber Results

The voltage tends towards 0.8 V after many hours inside the vacuum chamber, and from inspection is not expected to drop lower than this.

5.2 Simulation Results

There were around 200 simulations run for the study; therefore, only some graphs will be presented in figures in the report. All the results will, however, be presented in table format. The results are the deorbit times for each combination of satellite mass, orbit and deorbit device size.

5.2.1 Comparisons Between Balloon Size, Satellite Mass and Orbit Height

The first tests done were those of all three CubeSat sizes in three different orbits, mentioned in section 4.2.1 at mean solar activity conditions. The four balloon sizes mentioned in the same section were used to compare the results of the different deorbiting devices. A successful deorbit is shown in Figure 5.11, where a 12U satellite with a balloon device with radii of 0.1m $\times$ 0.2m $\times$ 0.2m deorbits from 480 km in 1174 days or 3.2 years. The deorbit was completed in dynamic solar activity conditions with the satellite having zero attitude changes over time.

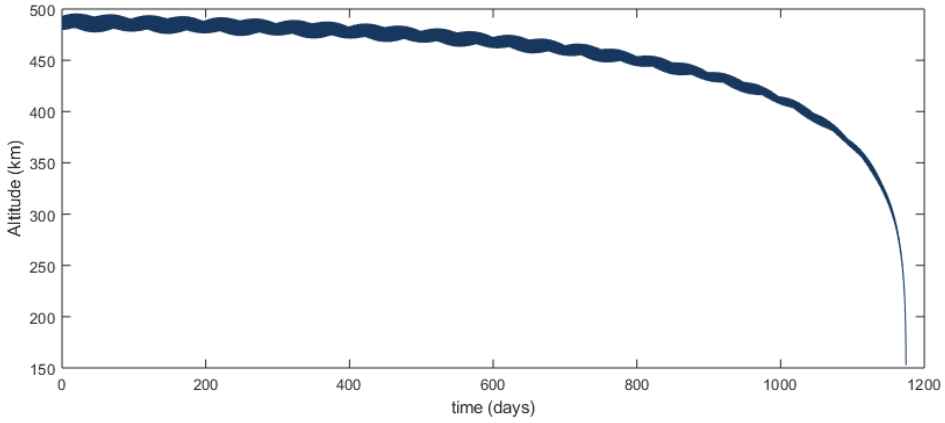


Figure 5.11: A Successful Deorbit

The durations for the satellites to deorbit without any additional deorbiting devices is given in Table 5.1. These are the control values of the satellites, done in dynamic solar activity conditions with constant attitude parameters. All the satellites masses are set to the heaviest allowable mass according to the CubeSat stander, that is, the 3U CubeSats have a mass of 4 kg, the 6U CubeSats have a mass of 8 kg and the 12U CubeSats have a mass of 16 kg. The deorbit times for the lowest orbit of 480 km will have to be ten times less than those of this control run, as specified in the project objectives in section 1.1. The simulations did not run for longer than 25 years in simulation time, because of the UN convention mentioned in section 2.2. As such those that took longer than 25 years to deorbit have "25+ years" noted as their deorbit times. The deorbit times of the two higher orbits, therefore, have to simply be less than 25 years to be considered an improvement to their unaided deorbit times and comply with the project objectives.

Table 5.1: Unaided Satellite Deorbit Times

Unaided	3U	6U	12U
480 km	1417 days (3.9 years)	1381 days (3.8 years)	2015 days (5.5 years)
600 − 800 km	25+ years	25+ years	25+ years
950 km	25+ years	25+ years	25+ years

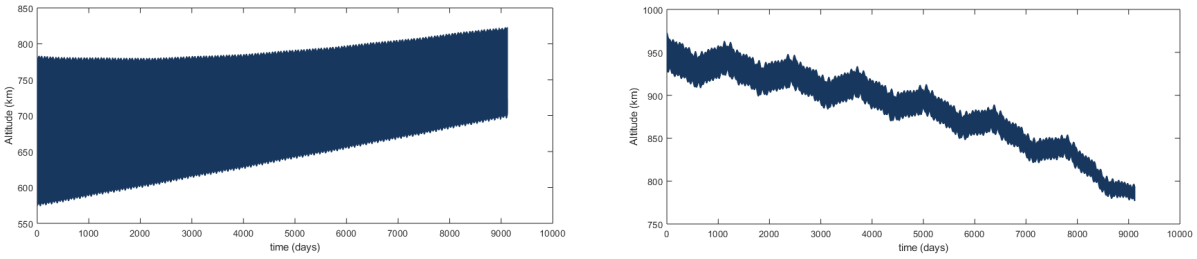
0.1m × 0.2m × 0.2m Balloon

Table 5.2 shows the deorbit times for the mean solar conditions of the first and smallest balloon tested. This balloon has radii of $a = 0.1m$, $b = 0.2m$ and $c = 0.2m$; with a the balloon x-radius, b the balloon y-radius and c the balloon z-radius.

Table 5.2: Results of the Balloon with Radii of 0.1 m, 0.2 m and 0.2 m in Mean Solar Conditions

0.1 m × 0.2 m × 0.2 m	3U	6U	12U
480 km	318 days (0.9 years)	595 days (1.6 years)	1200 days (3.3 years)
600 − 800 km	25+ years (down)	25+ years	25+ years
950 km	25+ years	25+ years	25+ years

The 3U elliptical orbit CubeSat has the word "down" added in brackets to the result. This descriptor is added because the deorbit device has an effect on the satellite's orbit and does start the deorbiting process. With the others, the device does not affect the CubeSat in the 25 years, and the satellite remains in its orbit. Figure 5.12 shows the comparison of a "25+ years" result compared to a "25+ years (down)" result.



(a) "25+ years" (b) "25+ years (down)"

Figure 5.12: Comparison Between Two Different "25+ years" Results

It is clear from these results in Table 5.2 that the smallest balloon size is not effective for the two higher orbits. It also does not improve the deorbit times of the 480 km orbit height CubeSats by a factor of ten and is therefore not a viable device at all. The minimum solar activity case will, therefore, also not work with this balloon size, as it will only take longer than in the mean solar activity case. The more realistic results are given in Table 5.3, where the changing atmospheric density model was used, starting at minimum conditions.

Table 5.3: Results of the Balloon with Radii of 0.1 m, 0.2 m and 0.2 m in Dynamic Solar Conditions

0.1 m × 0.2 m × 0.2 m	3U	6U	12U
480 km	604 days (1.6 years)	852 days (2.3 years)	1174 days (3.2 years)
600 − 800 km	2420 days (6.6 years)	7244 days (19.8 years)	25+ years
950 km	25+ years	25+ years	25+ years

This table shows that the 3U and 6U CubeSats in the elliptical orbit can be deorbited in less than 25 years with the smallest balloon, unlike in the constant mean solar activity. This is because the dynamic case would enter a time of maximum solar activity, with a higher atmospheric density than the mean solar activity case does not reach. Due to the dynamic activity starting at a minimum of the solar cycle (thus giving worst-case results), the satellites in the 480 km orbit still do not deorbit ten times faster than the unaided case, and those in the 950 km orbit do not deorbit within 25 years.

0.25m × 0.5m × 0.5m Balloon

The second balloon size, with radii of $a = 0.25m$, $b = 0.5m$ and $c = 0.5m$, was next to be simulated. The results of this balloon in constant mean solar activity are given in Table 5.4.

Table 5.4: Results of the Balloon with Radii of 0.25 m, 0.5 m and 0.5 m in Mean Solar Conditions

0.25 m × 0.5 m × 0.5 m	3U	6U	12U
480 km	44 days (0.12 years)	83 days (0.23 years)	162 days (0.4 years)
600 − 800 km	655 days (1.8 years)	1426 days (3.9 years)	3471 days (9.5 years)
950 km	25+ years	25+ years	25+ years

This balloon size is viable for 3U and 6U CubeSats in a 480 km orbit as it deorbits the satellite ten times faster than the control case with no deorbit device. However, this is not true for the 12U CubeSat in this orbit. This balloon size does work in the elliptical orbit, but not for the 950 km orbit. Mean solar activity is a good measure to use as an average-case result, but the worst-case will be determined with the minimum solar activity. These minimum solar activity results are presented in Table 5.5.

Table 5.5: Results of the Balloon with Radii of 0.25 m, 0.5 m and 0.5 m in Minimum Solar Conditions

0.25 m × 0.5 m × 0.5 m	3U	6U	12U
480 km	178 days (0.5 years)	301 days (0.8 years)	622 days (1.7 years)
600 − 800 km	922 days (2.5 years)	25+ years	25+ years
950 km	25+ years	25+ years	25+ years

It can be seen that the 3U CubeSat in the elliptical orbit is the only one that would deorbit within the specified objectives in minimum conditions. This result may be correct for constant minimum conditions, but the actual solar minimum lasts a few months; and the time before and after minimum with atmospheric densities lower than those in mean conditions only lasting just under three years. This duration of the minimum means that the results that are longer than 3 years are not accurate according to reality, nor are they an acceptable average-case model, because a minimum cycle would not last that long. The dynamic model results for this second balloon size are given in Table 5.6.

Table 5.6: Results of the Balloon with Radii of 0.25 m, 0.5 m and 0.5 m in Dynamic Solar Conditions

0.25 m × 0.5 m × 0.5 m	3U	6U	12U
480 km	146 days (0.4 years)	258 days (0.7 years)	430 days (1.2 years)
600 − 800 km	922 days (2.5 years)	1255 days (3.4 years)	1741 days (4.8 years)
950 km	25+ years (down)	25+ years	25+ years

While the satellites in the 480 km orbit no longer fulfil the requirements for deorbiting set out in the project objectives, the elliptical orbit CubeSats once again do fulfil the requirements; even with the dynamic case being the worst-case version thereof. This test is also the first time that a satellite in the 950 km orbit is affected by having a deorbiting device, even if it still does not deorbit within 25 years.

0.5m × 1.0m × 1.0m Balloon

The third balloon size was $a = 0.5m$, $b = 1.0m$ and $c = 1.0m$, double that of the previous balloon. This balloon's results as a deorbiting device are given in Table 5.7.

Table 5.7: Results of the Balloon with Radii of 0.5 m, 1.0 m and 1.0 m in Mean Solar Conditions

0.5 m × 1.0 m × 1.0 m	3U	6U	12U
480 km	12 days (0.03 years)	20 days (0.06 years)	40 days (0.11 years)
600 − 800 km	156 days (0.4 years)	294 days (0.8 years)	593 days (1.6 years)
950 km	3918 days (10.7 years)	25+ years (down)	25+ years

This balloon is much more successful in the objectives compared to the previous two balloons. It deorbits the satellites in the 480 km orbit more than a factor of ten times faster than the control case for all CubeSat sizes. It also successfully deorbits the satellites in the elliptical orbit within 25 years and is the first device to deorbit the 3U satellite in the 950 km orbit successfully. The mean solar activity case is promising but has to be verified against the constant minimum solar activity case. The worst-case scenario of this balloon is given in Table 5.8

Table 5.8: Results of the Balloon with Radii of 0.5 m, 1.0 m and 1.0 m in Minimum Solar Conditions

0.5 m × 1.0 m × 1.0 m	3U	6U	12U
480 km	43 days (0.12 years)	72 days (0.2 years)	142 days (0.4 years)
600 − 800 km	634 days (1.7 years)	1739 days (4.8 years)	4402 days (12.1 years)
950 km	25+ years (down)	25+ years	25+ years

As seen in Table 5.8, the minimum solar activity deorbits all the same CubeSats within the specified times, besides the 3U CubeSat in the furthest orbit. The 12U CubeSat in the elliptical orbit produced an unrealistic result, as a solar minimum would not last for 12 years. This is one of the results that are not accurate according to reality, and therefore should not be considered a viable one. To determine if the constant mean activity is accurate enough, the dynamic case for this balloon has to be analysed. Table 5.9 shows the dynamic case's results.

Table 5.9: Results of the Balloon with Radii of 0.5 m, 1.0 m and 1.0 m in Dynamic Solar Conditions

0.5 m × 1.0 m × 1.0 m	3U	6U	12U
480 km	37 days (0.1 years)	70 days (0.19 years)	134 days (0.4 years)
600 − 800 km	413 days (1.1 years)	614 days (1.7 years)	883 days (2.4 years)
950 km	1810 days (5 years)	3156 days (8.6 years)	25+ years (down)

The dynamic case has even better results in the more extended simulations that the mean case. This improvement is because the simulation includes the times of maximum solar conditions, which would make the atmospheric density higher and thus quicken the deorbiting process. This simulation shows that not just the 3U CubeSat in the highest orbit will deorbit with a balloon with the radii specified above, but that the 6U CubeSat

in this orbit will also deorbit. The mean solar activity average-case is, therefore, a more conservative case than the dynamic one for extended duration deorbits. The dynamic simulation also confirms the deorbit times from the lowest orbit of 480 km and the elliptical one.

1.0m × 2.0m × 2.0m Balloon

The final and largest balloon is once again double the size of the last one, with radii $a = 1.0m$, $b = 2.0m$ and $c = 2.0m$. The results of the mean solar activity simulation are in Table 5.10.

Table 5.10: Results of the Balloon with Radii of 1.0 m, 2.0 m and 2.0 m in Mean Solar Conditions

1.0 m × 2.0 m × 2.0 m	3U	6U	12U
480 km	2.7 days (0.01 years)	5.1 days (0.01 years)	9.9 days (0.03 years)
600 − 800 km	39 days (0.11 years)	74 days (0.2 years)	143 days (0.4 years)
950 km	541 days (1.5 years)	1496 days (4.1 years)	3477 days (9.5 years)

This balloon is the only one that works for all three orbits, deorbiting the CubeSats in the lowest orbit by a factor of ten times faster than the control case while also deorbiting the two in the higher orbits in less than 25 years. This simulation was, once again, verified against the constant minimum solar activity case and the results of that simulation are given in Table 5.11.

Table 5.11: Results of the Balloon with Radii of 1.0 m, 2.0 m and 2.0 m in Minimum Solar Conditions

1.0 m × 2.0 m × 2.0 m	3U	6U	12U
480 km	9.4 days (0.03 years)	17.8 days (0.05 years)	35 days (0.1 years)
600 − 800 km	158 days (0.4 years)	282 days (0.8 years)	564 days (1.5 years)
950 km	1778 days (4.9 years)	3937 days (9.5 years)	25+ years (down)

The minimum case is viable for all the deorbit times besides the 12U CubeSat in the 950 km orbit, but, once again, the minimum solar activity does not last as long as the total deorbit times for this highest orbit. The final simulation of this section is the dynamic simulation with this balloon. The results of this simulation are presented in Table 5.12.

Table 5.12: Results of the Balloon with Radii of 1.0 m, 2.0 m and 2.0 m in Dynamic Solar Conditions

1.0 m × 2.0 m × 2.0 m	3U	6U	12U
480 km	9.4 days (0.03 years)	17.7 days (0.05 years)	34 days (0.09 years)
600 − 800 km	149 days (0.4 years)	245 days (0.7 years)	391 days (1.1 years)
950 km	623 days (1.7 years)	1096 days (3 years)	1725 days (4.7 years)

This simulation, once more, verifies the mean solar conditions deorbit times. The deorbit times longer than three years yet again deorbited faster in the dynamic conditions due to the solar maximum activity sections in the dynamic simulations.

5.2.2 Dynamic Attitudes Versus Static Attitudes

The next set of tests were done using balloons as deorbiting devices on satellites with a changing yaw value, to see if it had any impact on the deorbit times. The simulations were only done on the three CubeSat sizes of the lowest orbit, that is the 480 km orbit. Figure 5.13 shows the positive yaw rate direction as golden arrows. The figure also depicts the radii of the balloon, with a being the short radius of the oblate spheroid. The orientation of the satellite is chosen as follows because many CubeSats are used for Earth Observation missions, with a camera lens facing nadir from the 1U face.

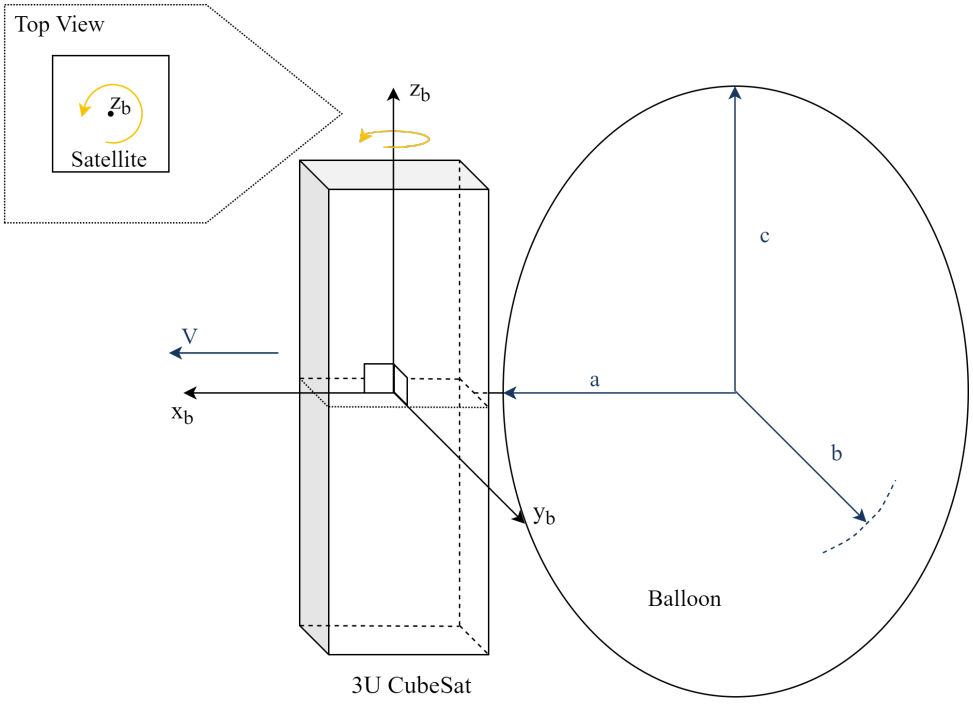


Figure 5.13: Entire System with a 3U CubeSat

The 6U and 12U set ups are shown in Figure 5.14. The balloons have been omitted for spacing, but they would be placed in the same manner as with the 3U CubeSats, behind the negative x face.

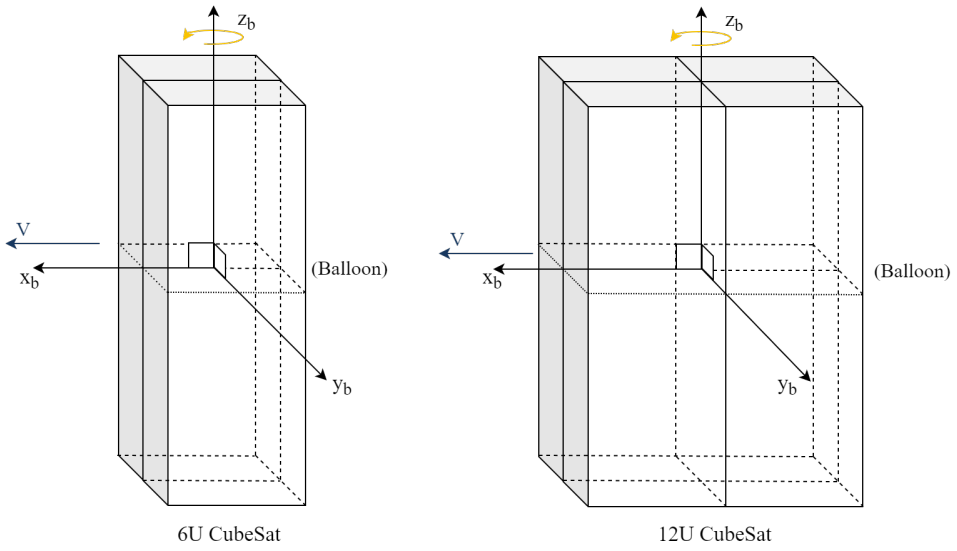


Figure 5.14: 6U and 12U CubeSat Orientations

A yaw around the satellite's z-axis and a pitch around its y-axis would have the same effect on the balloon's cross-sectional area, and as such, only a dynamic yaw motion was implemented. A roll around the z-axis would have no change in the balloon's cross-sectional area and was not implemented. The dynamic yaw motion was implemented using the following sine function:

$$\omega_z(t) = \omega_{zmax} \sin\left(\frac{\pi t}{4T_s} + \frac{\pi}{4}\right) \tag{5.1}$$

where ω_{max} is the maximum allowable yaw rate, t is the time, and T_s is the sample time. This formula was chosen as it represents the satellite yawing back-and-forth. The sine was shifted by $\frac{\pi}{4}$ so that the CubeSat does not only yaw to a certain point and back to the starting position. This shift allows the CubeSat to yaw away from the starting position, return back to it, then go beyond before returning. With no shift the balloon would yaw in the positive direction and return, and if a cosine function is used, the opposite would happen. This equation was implemented in the Simulink program's "Update Attitude" block, where the code can be found in Appendix D section D.2.2. For reference, all of the lowest orbit results from the previous section are summed up in Table 5.13.

Table 5.13: Summarised Results of the Balloon Device in a 480 km Orbit Height in Dynamic Solar Conditions

	3U	6U	12U
0.1 m × 0.2 m × 0.2 m	604 days (1.6 years)	852 days (2.3 years)	1174 days (3.2 years)
0.25 m × 0.5 m × 0.5 m	146 days (0.4 years)	258 days (0.7 years)	430 days (1.2 years)
0.5 m × 1.0 m × 1.0 m	37 days (0.1 years)	70 days (0.19 years)	134 days (0.4 years)
1.0 m × 2.0 m × 2.0 m	9.4 days (0.03 years)	17.7 days (0.05 years)	34 days (0.09 years)

The results of the simulations in static attitudes (time in days) were rounded to the nearest integer value unless the times were less than 20 days. The results from the dynamic attitude simulations were close to each other, especially the first two cases, and as such were rounded to the nearest single decimal place.

Maximum Yaw Rate of 1 Degree per Second

The first test had a maximum yaw rate value of 1 °/sec. The results from this test are given in Table 5.14

Table 5.14: Results of the Balloon When Yawing at up to 1 °/sec in Dynamic Solar Conditions

1 °/sec	3U	6U	12U
0.1 m × 0.2 m × 0.2 m	603.8 days	852 days	1174.4 days
0.25 m × 0.5 m × 0.5 m	145.6 days	258.1 days	429.8 days
0.5 m × 1.0 m × 1.0 m	37.4 days	70.4 days	134.1 days
1.0 m × 2.0 m × 2.0 m	9.4 days	17.7 days	34.3 days

This small yawing motion did not have any effect on the deorbit times of the CubeSats. They all took the same amount of time in full days to deorbit the satellites.

Maximum Yaw Rate of 5 Degrees per Second

The next simulation implemented satellites with maximum yaw rates of 5 °/sec. These results are presented in Table 5.15

Table 5.15: Results of the Balloon When Yawing at up to 5 °/sec in Dynamic Solar Conditions

5 °/sec	3U	6U	12U
0.1 m × 0.2 m × 0.2 m	606.1 days	855.3 days	1178.9 days
0.25 m × 0.5 m × 0.5 m	146.3 days	259.2 days	431.6 days
0.5 m × 1.0 m × 1.0 m	37.6 days	70.7 days	134.2 days
1.0 m × 2.0 m × 2.0 m	9.4 days	17.8 days	34.5 days

There was less than a day's difference in time with the shorter deorbit times, those of 100 days or less; however, those that are longer took a few days extra to deorbit. The difference is not yet significant, and those that were valid deorbit times for the specific CubeSat sizes are still valid for those that yaw in orbit.

Maximum Yaw Rate of 10 Degrees per Second

The results of the final yaw rate test are depicted in Table 5.16, where the CubeSats each could have a maximum yaw rate of 10 °/sec.

Table 5.16: Results of the Balloon When Yawing at up to 10 °/sec in Dynamic Solar Conditions

10 °/sec	3U	6U	12U
0.1 m × 0.2 m × 0.2 m	615.4 days	867 days	1194.4 days
0.25 m × 0.5 m × 0.5 m	149.7 days	264.7 days	439.3 days
0.5 m × 1.0 m × 1.0 m	38.5 days	72.5 days	137.9 days
1.0 m × 2.0 m × 2.0 m	9.7 days	18.2 days	35.3 days

Even though the differences are more substantial in this test, the deorbit times of those that were ten times faster than the unaided case, are still valid within the project's objectives. It is reasonable to believe that the satellites will not be yawing any faster than 10 degrees per second. Therefore, if a CubeSat with a balloon device on it starts to yaw in orbit, it will not significantly impact the deorbit times of the satellite. This is because the drag area of the balloon is not changed by much and is on average the same while deorbiting. The fastest angular rate case will cause the smallest average drag area, hence it taking slightly longer to deorbit.

5.2.3 Using a Sail Device Instead of a Balloon Device

Two MATLAB simulations were done using a sail instead of a balloon as the deorbiting device for the CubeSats. The tests had dynamic solar conditions implemented at the lowest orbit height and did not include any rotation of the satellites. The sail thickness is measured in the x-direction, the length is in the y-direction, and the height is in the z-direction. The thickness of the sail was chosen at 1 mm for all the simulations such that it can be negligible compared to the length and height. The orientations of the three CubeSat sizes with the sails is shown in Figure 5.15.

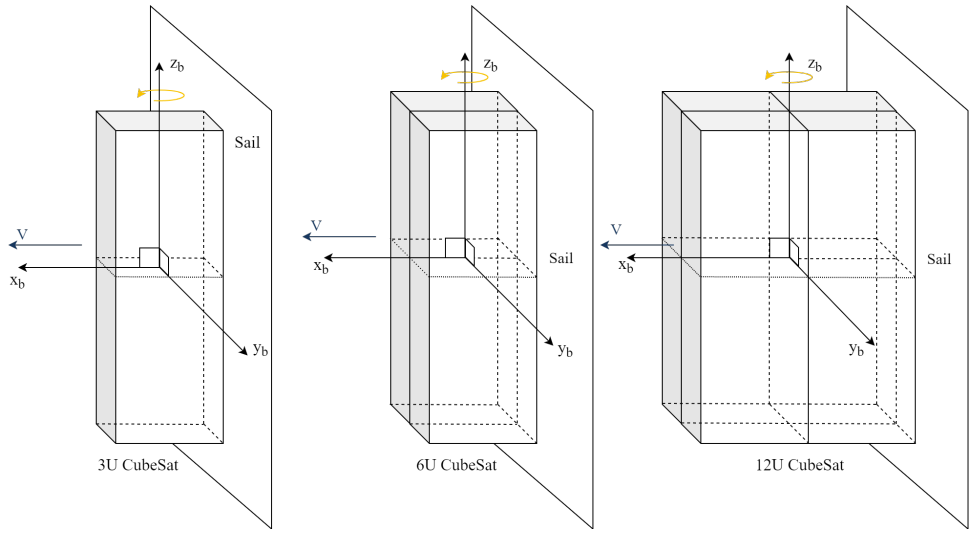


Figure 5.15: 3U, 6U and 12U System with a Sail

Once more, for reference, the results of the 480 km orbit simulations are summarised in Table 5.17

Table 5.17: Summarised Results of the Balloon Device in a 480 km Orbit Height in Dynamic Solar Conditions

	3U	6U	12U
0.1 m × 0.2 m × 0.2 m	604 days (1.6 years)	852 days (2.3 years)	1174 days (3.2 years)
0.25 m × 0.5 m × 0.5 m	146 days (0.4 years)	258 days (0.7 years)	430 days (1.2 years)
0.5 m × 1.0 m × 1.0 m	37 days (0.1 years)	70 days (0.19 years)	134 days (0.4 years)
1.0 m × 2.0 m × 2.0 m	9.4 days (0.03 years)	17.7 days (0.05 years)	34 days (0.09 years)

Sail Length and Height Double Balloon y- and z-radii

The sail tests were implemented to compare the results to the balloon device's results. Initially, the length and height of the sails were chosen to be double the radii of the balloons to have similar projected areas as the balloons. These chosen dimensions mean that the surface area of the sail would be slightly larger than the yz-cross-sectional area of the balloon. This sail configuration yielded the results given in Table 5.18.

Table 5.18: Results of the First Sail Simulations

	3U	6U	12U
0.4 m × 0.4 m	568 days (1.6 years)	827 days (2.3 years)	1176 days (3.2 years)
1.0 m × 1.0 m	122 days (0.3 years)	224 days (0.6 years)	391 days (1.1 years)
2.0 m × 2.0 m	31 days (0.08 years)	58 days (0.16 years)	112 days (0.3 years)
4.0 m × 4.0 m	7.6 days (0.02 years)	14.5 days (0.04 years)	28 days (0.08 years)

As it is to be expected, the CubeSats deorbit slightly faster than the corresponding balloons, because of the larger drag affected areas. This accelerated deorbit time is due to the area affected by drag being more extensive than that of the balloons.

Sail Length and Height with Projected Area Similar to Balloon

The lengths and heights of the sails were modified such that the surface area of the sails is more or less the same as the yz-cross-sectional area of the balloons, to have more comparable results. This test provides a better contrast of the sail deorbit times versus the balloon deorbit times, and the results are presented in Table 5.19.

Table 5.19: Results of the Second Sail Simulations

	3U	6U	12U
0.35 m × 0.35 m	670 days (1.8 years)	956 days (2.6 years)	1352 days (3.7 years)
0.89 m × 0.89 m	153 days (0.4 years)	275 days (0.8 years)	464 days (1.3 years)
1.77 m × 1.77 m	39 days (0.11 years)	74 days (0.2 years)	143 days (0.4 years)
3.5 m × 3.5 m	10 days (0.03 years)	18.9 days (0.05 years)	37 days (0.1 years)

It is interesting to note that the sail devices deorbit the CubeSats slightly slower than the balloon devices of the same size. This difference could be due to the SRP force acting on the side of the balloons and thus speeding up the deorbit process slightly. The difference is most noticeable with the longer duration deorbits, and even the shorter duration ones take more time to deorbit.

5.2.4 STELA Results Compared to MATLAB Results

STELA has the drag and reflective areas as parameters. Thus, the areas were not put in as a product of length and height - as STELA only works with sails - but just as the total surface area in m^2. The areas used were the same as those of the balloon devices. The simulations were run in both mean constant solar conditions and dynamic solar conditions for the 480 km orbit height CubeSats. STELA outputs the time taken to

deorbit the satellites in years rounded to two decimal places, instead of days. As such, determining the number of days to deorbit is not as exact as the MATLAB program.

Mean Constant Solar Conditions

The summarised mean solar activity results from section 5.2.1 for the 480 km orbit are given in Table 5.20 for reference.

Table 5.20: Summarised Results of the Balloon Device in a 480 km Orbit Height in Mean Solar Conditions

	3U	6U	12U
0.1 m × 0.2 m × 0.2 m	318 days (0.9 years)	595 days (1.6 years)	1200 days (3.3 years)
0.25 m × 0.5 m × 0.5 m	44 days (0.12 years)	83 days 0.23 years)	162 days (0.4 years)
0.5 m × 1.0 m × 1.0 m	12 days (0.03 years)	20 days (0.05 years)	40 days (0.11 years)
1.0 m × 2.0 m × 2.0 m	2.7 days (0.01 years)	5.1 days (0.01 years)	9.9 days (0.03 years)

The results from the STELA simulation using a mean constant solar activity are presented in Table 5.21.

Table 5.21: STELA Results of CubeSats in a 480 km Orbit with Mean Constant Solar Activity

	3U	6U	12U
0.126 m^2 (0.2×0.2×π)	256 days (0.7 years)	515 days (1.41 years)	1044 days (2.86 years)
0.785 m^2 (0.5×0.5×π)	40 days (0.11 years)	71 days (0.2 years)	146 days (0.4 years)
3.142 m^2 (1.0×1.0×π)	11.7 days (0.03 years)	22 days (0.06 years)	38 days (0.1 years)
12.57 m^2 (2.0×2.0×π)	3.9 days (0.01 years)	7 days (0.02 years)	11 days (0.03 years)

The results are very similar to the MATLAB program implemented for this study and those from STELA. The most considerable differences are found with the smallest size devices. The MATLAB program is more conservative with the deorbit times, which is not a bad thing, as it is probably better to overestimate the time it takes for a satellite to deorbit than to underestimate it. The MATLAB sail simulations were even slower than the balloon simulations, which makes the differences between MATLAB and STELA slightly more significant for the sail devices.

Dynamic Solar Conditions

A recap of the MATLAB program's results of the lowest orbit in dynamic solar conditions is given in Table 5.22 again for reference.

Table 5.22: Summarised Results of the Balloon Device in a 480 km Orbit Height in Dynamic Solar Conditions

	3U	6U	12U
0.1 m × 0.2 m × 0.2 m	604 days (1.6 years)	852 days (2.3 years)	1174 days (3.2 years)
0.25 m × 0.5 m × 0.5 m	146 days (0.4 years)	258 days (0.7 years)	430 days (1.2 years)
0.5 m × 1.0 m × 1.0 m	37 days (0.1 years)	70 days (0.19 years)	134 days (0.4 years)
1.0 m × 2.0 m × 2.0 m	9.4 days (0.03 years)	17.7 days (0.05 years)	34 days (0.09 years)

STELA uses a pre-loaded file for the dynamic solar conditions, and the atmospheric density changes depending on what time the simulation has its epoch. The exact time of the minimum point in the solar cycle is unknown and was estimated after a few simulations showing the solar flux levels. Table 5.23 shows the STELA results in dynamic solar activity, starting from around the minimum point in the cycle.

Table 5.23: STELA Results of CubeSats in a 480 km Orbit with Dynamic Solar Activity

	3U	6U	12U
0.126 m^2 (0.2×0.2×π)	595 days (1.63 years)	792 days (2.17 years)	1128 days (3.09 years)
0.785 m^2 (0.5×0.5×π)	226 days (0.62 years)	332 days (0.91 years)	467 days (1.28 years)
3.142 m^2 (1.0×1.0×π)	73 days (0.2 years)	142 days (0.39 years)	212 days (0.58 years)
12.57 m^2 (2.0×2.0×π)	23 days (0.06 years)	36 days (0.1 years)	66 days (0.18 years)

The results from STELA vary somewhat from those of MATLAB, with some taking twice as long to deorbit (the shorter deorbit times) and others being near-identical (the more extended deorbit times). A deorbit time under 100 days might be challenging to calculate precisely, which could be the reason for these differences.

Chapter 6

Conclusion and Recommendations

A concise overview of the project objectives achieved is given at the start of this chapter which concludes the thesis. The chapter ends with recommendations for further work that could be applied to improve the project. The project objectives are stated below again for reference.

Project Objectives

The first objective of this project is to design, make and test a small, low mass and volume device to deploy at End-of-Life (EoL) for a CubeSat to aid in the deorbiting thereof. The prototype device must be tested in vacuum conditions to ensure it can be operated in space. It must accelerate the deorbit rate by at least a factor of ten from an initial altitude of at least 700 km. The prototype must be able to work regardless of the CubeSat's attitude and have a self-sustaining power source for deployment. The second and final objective of this project is to undergo simulation studies to support the choice of the prototype. The best device will be selected for the specific CubeSat size and orbit height based on the simulation results.

6.1 Summary and Results Discussion

The prototype system works well. Steps were taken to use low-power components. The system can open and close the valve with commands as well as check the level of the strain gauge voltage and open the valve again if necessary. The system also uses small components with the largest being the microcontroller. If implemented on a mission, only the ATtiny416 chip will be used and not the entire development board. The balloon can be folded into tiny sizes, with the store-bought one being able to fit inside a 1U space while only being 5 mm thick. The volume of air needed to fill the balloon can be compressed into a small container. Only 100 ml of air at atmosphere was needed to fill at 6.6 ℓ balloon in the vacuum-like conditions of the vacuum chamber. The system runs off of its battery and can go into an ultra-low-power sleep mode while the satellite is on its mission.

The simulations produced realistic results compared to the deorbiting analysis program STELA, albeit slightly more conservative. The simulations work for satellites up to 1000 km in orbit height. The two smaller balloon sizes do not deorbit the CubeSats in the lowest orbit of 480 km by a factor of ten times faster than what they would deorbit in being unaided by a deorbiting device. The second smallest balloon, with radii $a = 0.25m$, $b = 0.5m$ and $c = 0.5m$, does deorbit the CubeSats under 25 years but not those in the highest orbit of 950 km. The third balloon size, with radii $a = 0.5m$, $b = 1.0m$ and

$c = 1.0m$, successfully deorbits all the CubeSats besides the 12U one in the highest orbit, according to the dynamic solar condition results. The largest balloon includes this 12U CubeSat in the successful deorbits meaning all the satellites deorbit effectively with this balloon. However, it is unnecessary to put such a giant balloon on the lower orbit CubeSats. The largest balloon is only recommended for orbits higher than 800 km, while the third balloon is the recommended balloon size for the orbits below that. The mean solar conditions and dynamic solar conditions give similar results in terms of viable deorbit times. The mean simulations caused faster deorbit times for shorter duration deorbits but is acceptable to use in general cases, as it is less computationally expensive than the dynamic cases.

From the MATLAB simulations, it can be seen that the sails tend to take slightly longer to deorbit the CubeSats than the balloons do. This difference could be due to SRP aiding the balloon more by being able to reflect off the side of the balloon. If the satellite is tumbling out of control when the balloon is deployed, it will correct to a certain extent. If the centre-of-pressure is behind the centre-of-mass the satellite will be passively stabilised over time, when there is any energy loss from atmospheric drag for example. As such, the balloon would move to the back and the satellite in front, due to the balloon being the centre-of-pressure and the satellite the centre-of-mass. The satellite may have a yawing or pitching motion. This changing attitude was tested in the simulations by implementing a dynamic yawing motion on the CubeSat (pitching and yawing would have the same effect on the balloon). The simulations show that the attitude of the satellite does not make a significant difference in the deorbit time, only slowing the deorbit times by a few days at the fastest maximum yaw rate of 10 °/sec.

6.2 Recommendations and Future Work

Further investigation into why the strain gauge voltage continues to drop in vacuum-like conditions is recommended for future work. Especially since that did not happen during the atmosphere-level tests. Using a different strain gauge or a better quality balloon may yield different results.

The study was done with the balloon and sail made entirely of the JPL solar sail material with the assumption that the satellite could no longer be controlled. However, if the satellite is still completely functional at the end of its mission, and is chosen to deorbit at this point instead of waiting for the satellite's EoL, the balloon could be designed so that half of the material will work as a solar sail while the other half does not. The satellite could then be controlled to rotate so that the material faces the sun when the satellite approaches the sun and turns around when it goes the other way so that the non-reflective material then faces the sun. This change in attitude would cause the satellite to slow down even faster as the SRP would "push" the satellite in the opposite direction of movement when approaching the sun and not cause the opposite effect when the satellite moves away from the sun. This approach could be especially useful for CubeSats in higher orbits such as upwards from 800 km.

Bibliography

[1] Clark, S.: A chat with Bob Twiggs, father of the CubeSat. *Spaceflight Now*, March 2014. Available at: `https://spaceflightnow.com/news/n1403/08cubesats/#.WF-8ulN95jE`

[2] Kulu, E.: Nanosats database. 2019. Accessed October 2019, last updated 10/06/2019. Available at: `https://www.nanosats.eu/`

[3] Bellardo, J.: THE CUBESAT PROGRAM. Year Unknown. Accessed September 2019. Available at: `http://www.cubesat.org/about`

[4] Loff, S. (ed.): CubeSats Overview. February 2018. Accessed September 2019. Available at: `https://www.nasa.gov/mission_pages/cubesats/overview`

[5] Kulu, E.: What is a CubeSat & other picosatellites. 2019. Accessed September 2019. Available at: `https://www.nanosats.eu/cubesat`

[6] Mabrouk, E. (ed.): What are SmallSats and CubeSats? February 2015. Accessed October 2019. Available at: `https://www.nasa.gov/content/what-are-smallsats-and-cubesats`

[7] Puig-Suari, J. and Twiggs, B.: CubeSat Design Specification. Design specification, California Polytechnic State University, February 2014. Revision 13, by Mehrparvar, A.

[8] g3wgm (Username): P-Pod orbital Deployer. December 2012. Accessed October 2019. Available at: `https://amsat-uk.org/p-pod-orbital-deployer/`

[9] RH: Payload Specification for 3U, 6U, 12U and 27U. Design specification, Planetary Systems Corporation, August 2016. Revision D, only initials given for creators.

[10] Wertz, J., Everett, D. and Puschell, J. (eds.): *Space Mission Engineering: The New SMAD*. Space Technology Library and Microcosm Press, 2011.

[11] Kulu, E.: Selected Upcoming CubeSat Missions. 2019. Accessed October 2019, last updated 31/12/2018. Available at: `https://www.nanosats.eu/tables#missions`

[12] Klesh, A. and Krajewski, J.: MarCO: CubeSats to Mars in 2016. In: *Proceedings of the 29th Annual AIAA/USU Conference on Small Satellites*. Logan, Utah, USA, August 2015. Available at: `https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=3180&context=smallsat`

[13] Asmar, S. and Matousek, S.: Mars Cube One (MarCO) Shifting the Paradigm in Relay Deep Space Operations. In: *Proceedings of the 14th International Conference on Space Operations (SpaceOps 2016)*. Daejeon, Korea, May 2016. Available at: `http://arc.aiaa.org/doi/pdf/10.2514/6.2016-2483`

[14] Foust, J.: Mars cubesats fall silent. February 2019. Accessed October 2019.
Available at: `https://spacenews.com/mars-cubesats-fall-silent/`

[15] Howell, E.: CubeSats: Tiny Payloads, Huge Benefits for Space Research. June 2018.
Available at: `https://www.space.com/34324-cubesats.html`

[16] Anon.: IADC Space Debris Mitigation Guidelines. Guideline, Inter-Agency Space Debris Coordination Committee, September 2007. Revision 1, IADC-02-01.

[17] Gleghorn, G., Asay, J., Atkinson, D., Flury, W., Johnson, N., Kessler, D., Knowles, S., Rex, Dietrich Toda, S. and Veniaminov, S.: *Orbital Debris: A Technical Assessment*. National Academies Press, Washington, DC, USA, 1995. ISBN 9780309051255.
Available at: `https://ntrs.nasa.gov/search.jsp?R=19950022431`

[18] Orbital Debris Program Office: Astromaterials Research & Exploration Science. 2019. Accessed October 2019.
Available at: `https://orbitaldebris.jsc.nasa.gov/`

[19] Liou, J.-C. and Johnson, N.L.: Risks in space from orbiting debris. *Science*, vol. 311, pp. 340 – 341, January 2006.

[20] Kessler, D.J. and Cour-Palais, B.G.: Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research: Space Physics*, vol. 83, no. A6, pp. 2637–2646, 1978.

[21] Scientific and Technical Subcommittee of the United Nations Committee on the Peaceful uses of Outer Space (eds.): Technical report on space debris. Tech. Rep., United Nations, New York, USA, 1999.

[22] Anon.: The Threat of Orbital Debris and Protecting NASA Space Assets from Satellite Collisions. April 2009.
Available at: `http://images.spaceref.com/news/2009/ODMediaBriefing28Apr09-1.pdf`

[23] Federal Communications Commission: In the matter of mitigation of orbital debris. Rule-making, Federal Communications Commission, Washington, D.C., USA, March 2002. DA/FCC #: FCC-02-80.
Available at: `https://www.fcc.gov/document/matter-mitigation-orbital-debris`

[24] Mark, C.P. and Kamath, S.: Review of Active Space Debris Removal Methods. *Space Policy*, vol. 47, pp. 194 – 206, December 2018. ISSN 0265-9646.

[25] Hakima, H. and Emami, M.R.: Assessment of active methods for removal of leo debris. *Acta Astronautica*, vol. 144, pp. 225 – 243, 2018. ISSN 0094-5765.

[26] Werner, D.: ESA to investigate links between debris removal and satellite servicing. March 2018.
Available at: `https://spacenews.com/esa-to-investigate-links-between-debris-\\removal-and-satellite-servicing/`

[27] O'Hare, R.: British space junk mission to test nets, sails and HARPOONS to catch dangerous orbital debris. July 2016. Last updated 05/07/2016.
Available at: `https://www.dailymail.co.uk/sciencetech/article-3674977/`

[28] Phipps, C.R., Baker, K.L., Libby, S.B., Liedahl, D.A., Olivier, S.S., Pleasance, L.D., Rubenchik, A., Trebes, J.E., George, E.V., Marcovici, B., Reilly, J.P. and Valley, M.T.: Removing orbital debris with lasers. *Advances in Space Research*, vol. 49, pp. 1283 – 1300, February 2012.

[29] Aslanov, V. and Yudintsev, V.: Dynamics of large space debris removal using tethered space tug. *Acta Astronautica*, vol. 91, pp. 149 – 156, 2013. ISSN 0094-5765.

[30] Takahashi, K., Charles, C., Boswell, R.W. and Ando, A.: Demonstrating a new technology for space debris removal using a bidirectional plasma thruster. *Scientific Reports*, vol. 8, no. 14417, September 2018. ISSN 2045-2322.

[31] Vallado, D.A. and McClain, W.D.: *Fundamentals of Astrodynamics and Applications*. 2nd edn. Microcosm Press, 2001.

[32] Canuto, E., Novara, C., Massotti, L., Carlucci, D. and Perex-Montenegro, C.: *Spacecraft Dynamics and Control The Embedded Model Control Approach*. Butterworth-Heinemann, Elsevier, Oxford, UK, 2018.

[33] Steyn, W.H.: Satellite systems 813. 2018. Unpublished lecture notes.

[34] Bhowmik, P. and Nandy, D.: Prediction of the strength and timing of sunspot cycle 25 reveal decadal-scale space environmental conditions. *Nature Communications*, vol. 9, no. 1, p. 5209, 2018. ISSN 2041-1723.
Available at: `https://doi.org/10.1038/s41467-018-07690-0`

[35] NASA's Community Coordinated Modelling Center: U.S. Standard Atmosphere 1976. . Accessed September 2019.
Available at: `https://ccmc.gsfc.nasa.gov/modelweb/atmos/us_standard.html`

[36] NASA's Community Coordinated Modelling Center: Standard Jacchia Reference Atmosphere 1977. . Accessed September 2019.
Available at: `https://ccmc.gsfc.nasa.gov/modelweb/atmos/jacchia.html`

[37] Picone, J.M., Hedin, A.E., Drob, D.P. and Aikin, A.C.: Nrlmsise-00 empirical model of the atmosphere: Statistical comparisons and scientific issues. *Journal of Geophysical Research: Space Physics*, vol. 107, no. A12, pp. SIA 15–1–SIA 15–16, 2002. `https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2002JA009430`.
Available at: `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2002JA009430`

[38] United States Patent and Trademark Office: Trademark Status & Document Retrieval (TSDR). Year Unknown. Accessed September 2019.
Available at: `https://tsdr.uspto.gov/#caseNumber=85741076&caseType=SERIAL_NO&searchType=statusSearch`

[39] Doornbos, E.: *Thermospheric Density and Wind Determination from Satellite Dynamics*. Ph.D. thesis, Delft University of Technology, March 2011.

[40] Hedin, A.: Msis model (1986). *Planetary and Space Science*, vol. 40, no. 4, pp. 555 – 556, 1992. ISSN 0032-0633.
Available at: `http://www.sciencedirect.com/science/article/pii/0032063392902097`

[41] Hedin, A.E.: Extension of the msis thermosphere model into the middle and lower atmosphere. *Journal of Geophysical Research: Space Physics*, vol. 96, no. A2, pp. 1159–1172, 1991.

[42] NASA's Community Coordinated Modelling Center: NRLMSISE-00 Model 2001. . Accessed September 2019.
Available at: `https://ccmc.gsfc.nasa.gov/modelweb/atmos/nrlmsise00.html`

[43] Lakdawalla, E.: #agu17: Spherical harmonics, gravity, and the depth of winds at jupiter. December 2017.
Available at: `https://www.planetary.org/blogs/emily-lakdawalla/2017/1220-agu17-spherical-harmonics.html`

[44] McInnes, C.: *Solar sailing: Technology, dynamics and mission applications.* 1st edn. Springer, 1999.

[45] Steyn, W.H. and Lappas, V.: Cubesat solar sail 3-axis stabilization using panel translation and magnetic torquing. *Aerospace Science and Technology*, vol. 15, no. 6, pp. 476 – 485, 2011. ISSN 1270-9638.

[46] *ATtiny416 Xplained Nano.* Microchip, October 2017. Revision A.

[47] *ATtiny416/816 AVR Microcontroller with Core Independent Peripherals and picoPower Technology.* Microchip, January 2017. Revision A.

[48] The Lee Company: *Electro-Fluidic Systems Handbook.* 9th edn. Printed in the USA, publisher not provided, Westbrook, Conneticut, USA, 2017.

[49] *A4973 Full-Bridge PWM Motor Driver.* Allegro Microsystems Inc., October 2011. Preliminary Revision.

[50] *INA333 Micro-Power (50μA), Zero-Drift, Rail-to-Rail Out Instrumentation Amplifier.* Texas Instruments, July 2008. Revision C, December 2015.

[51] Dismukes, K.: Definition of two-line element set coordinate system. September 2011.
Available at: `https://spaceflight.nasa.gov/realdata/sightings/SSapplications/Post/JavaSSOP/SSOP_Help/tle_def.html`

[52] French Space Agency (CNES): STELA. Year Unknown.
Available at: `https://logiciels.cnes.fr/en/content/stela`

Appendices

Appendix A

Atmel Studio Code

The code in this appendix is the code that was used to control the hardware, that is read in INA values and control the valve, written in Atmel Studio 7.0. Most of the comments have been moved above their commands or statements for aesthetic purposes.

```
1   /*
2   * Valve_control.c
3   *
4   * Created: 2018/09/20 15:53:30
5   * Author : 18199704
6   * Note: The maximal possible delay when using _delay_ms is (262.14 ms /
    F_CPU) in MHz which equals 16ms
7   */
8
9   #define F_CPU 16000000UL        //16 MHz clock speed
10
11  //for port A
12  #define closedLED   PIN4_bm
13  #define openLED     PIN5_bm
14  #define adcIn       PIN7_bm
15
16  //for port B
17  #define phase       PIN0_bm
18  #define nEnable     PIN1_bm
19  #define TxD         PIN2_bm
20  #define RxD         PIN3_bm
21  #define button      PIN4_bm
22  #define LED         PIN5_bm
23
24  #include <avr/io.h>
25  #include <util/delay.h>
26  #include <avr/interrupt.h>
27  #include <stdbool.h>
28  #include <stdlib.h>
29  #include <stdio.h>
30  #include <string.h>
31
32  volatile uint16_t count1 = 0;   //second counter1
33  volatile uint16_t count2 = 0;   //second counter2
34  bool valveOpen = false;         //valve state flag
35  bool prevFilled = false;        //valve has been made full flag
36
37  void clkInit()
38  {
39      //16Mhz internal oscillator (says 20 but for ATTiny416 it's 16MHz)
40      CLKCTRL.MCLKCTRLA = CLKCTRL_CLKSEL_OSC20M_gc;
41      //select 16MHz from internal oscillator
42      FUSE.OSCCFG = FREQSEL_16MHZ_gc;
43      //enable pre-scaler of 4 for peripheral clock
44      CLKCTRL.MCLKCTRLB = CLKCTRL_PDIV_4X_gc | CLKCTRL_PEN_bm;
45  }
46
```

```
47  void usartInit()
48  {
49      unsigned long fBaud = 19200;
50      uint16_t baudReg = ((double)((64*(F_CPU))/(16*fBaud*55.681)))*(10^6);
51      USART0.BAUDL = (uint8_t)(baudReg & 0xFF);
52      USART0.BAUDH = (uint8_t)(baudReg >> 8);
53
54      //set mode and frame format: mode, parity, stop bit, data size
55      USART0.CTRLC = USART_CMODE_ASYNCHRONOUS_gc | USART_PMODE_DISABLED_gc |
    USART_SBMODE_1BIT_gc | USART_CHSIZE_8BIT_gc;
56
57      //enable receive and transmit
58      USART0.CTRLB = USART_RXEN_bm | USART_TXEN_bm;
59  }
60
61  void timerInit()
62  {
63      //TOP value such that count increases every second
64      TCA0.SINGLE.PER = 0x20;
65      //count up
66      TCA0.SINGLE.CTRLECLR = TCA_SINGLE_DIR_UP_gc;
67      //pre-scaler of 1024 chosen, gives a 15.625kHz timer, total delay 4.19
    seconds, enable timer
68      TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc | TCA_SINGLE_ENABLE_bm;
69      //overflow timer interrupts enabled
70      TCA0.SINGLE.INTCTRL = TCA_SINGLE_OVF_bm;

71  }
72
73  //ISR below occurs when TCA CNT reaches TOP
74  ISR(TCA0_OVF_vect)
75  {
76      count1++;
77      if (count1 == 100)
78      {
79          count1 = 0;
80          PORTB.OUTTGL = LED;
81      }
82      count2++;
83
84      TCA0.SINGLE.INTFLAGS = TCA_SINGLE_OVF_bm;   //reset interrupt flag
85  }
86
87
88  void adcInit()
89  {
90      //set internal voltage Vref to 1.5V
91      VREF.CTRLA = VREF_ADC0REFSEL_1V5_gc;
92      //select internal voltage as Vref, peripheral clock (fcpu/4) divide by
    16 - 250kHz
93      ADC0.CTRLC = ADC_REFSEL_INTREF_gc | ADC_PRESC_DIV16_gc;
94      //10 bit resolution, enable free run
95      ADC0.CTRLA = ADC_RESSEL_10BIT_gc | ADC_FREERUN_bm;
96      //select AIN7 (PA7) as analog input
97      ADC0.MUXPOS = ADC_MUXPOS_AIN7_gc;
98      //enable adc
99      ADC0.CTRLA |= ADC_ENABLE_bm;
100 }
101
102
103 void openValve()
104 {
105     int i;
106     //green LED on, red LED off
107     PORTA.OUTTGL = openLED | closedLED;
108     //phase high and enable low to open valve
109     PORTB.OUTCLR = nEnable;
110     PORTB.OUTSET = phase;
111     for (i = 0; i < 10; i++) _delay_ms(10);
112     //enable high to disable h-bridge
```

```
113         PORTB.OUTSET = nEnable;
114         valveOpen = true;
115         count2 = 0;
116         //set flag for valve being opened before
117     //  if (firstOpen) firstOpen = false;
118     }
119
120     void closeValve()
121     {
122         int i;
123         //red LED on, green LED off
124         PORTA.OUTTGL = closedLED | openLED;
125         //phase low and enable low to close valve
126         PORTB.OUTCLR = phase | nEnable;
127         for (i = 0; i < 10; i++) _delay_ms(10);
128         //enable high to disable h-bridge
129         PORTB.OUTSET = nEnable;
130         valveOpen = false;
131     }
132
133     void boardSetup()
134     {
135         //1 for output, 0 for input
136         PORTA.DIRCLR = adcIn;
137         //PA4, PA5: outputs
138         PORTA.DIRSET = closedLED | openLED;
139         //PA4, PA5: low
140         PORTA.OUTCLR = closedLED | openLED;
141         //enable internal pull-up resistor
142         PORTB.PIN4CTRL = PORT_PULLUPEN_bm;
143         //PB3 input
144         PORTB.DIRCLR = RxD;
145         //PB0, PB1, PB4, PB2: outputs
146         PORTB.DIRSET = phase | nEnable | LED | TxD;
147         //PB0 (phase), PB4(LED): low.
148         PORTB.OUTCLR = phase | LED;
149         //PB1 (nEnable) is active low
150         PORTB.OUTSET = nEnable | TxD;
151     }
152
153     int main(void)
154     {
155         int i;
156         boardSetup();
157
158         //peripheral initialisations
159         clkInit();
160         usartInit();
161         timerInit();
162         adcInit();
163
164         //double check valve is closed
165         PORTB.OUTCLR = nEnable | phase;      //PB1, PB0: low
166         for (i = 0; i < 10; i++) _delay_ms(10);
167         PORTB.OUTSET = nEnable;              //enable high to disable h-bridge
168         PORTA.OUTSET = closedLED;            //red LED on
169
170         for (i = 0; i < 15; i++) _delay_ms(15);
171
172         //safety
173         while(!(PORTB.OUT & nEnable)) {
174             PORTB.OUTSET |= nEnable;
175         }
176
177         ADC0.COMMAND = ADC_STCONV_bm;        //start converting
178
179         sei();                               //enable global interrupts
180
181
182         uint8_t uartIn = 0x00;
```

```
183         uint16_t inaResult = 0x00;
184         double sgVolt;
185         double refVol = 1.50;
186         uint16_t Tclose = 65535;
187         char strResult[10] = "NULL";
188 
189         while (1)
190         {
191             //Get the converted result from the SG in-amp
192             inaResult = ADC0.RES;
193 
194             if (USART0.STATUS & USART_RXCIF_bm)
195             {
196                 //Read USART instruction
197                 uartIn = USART0.RXDATAL;
198             }
199 
200             if (uartIn != 0x00)
201             {
202                 //between 1 and 9 for number of seconds open
203                 if ((uartIn > '0') && (uartIn <= '9') && !valveOpen)
204                 {
205                     //ascii to int
206                     Tclose = (uartIn - 48)*100;
207                     openValve();
208                 }
209 
210                 //'o' to open indefinitely
211                 if (uartIn == 'o' && !valveOpen)
212                 {
213                     openValve();
214                 }
215 
216                 //'c' for close
217                 if (uartIn == 'c')
218                 {
219                     closeValve();
220                 }
221                 uartIn = 0x00;
222             }
223 
224             //open valve with button
225             if(!(PORTB.IN & button) && !valveOpen)
226             {
227                 for (i = 0; i < 10; i++) _delay_ms(15);     //debounce
228                 openValve();
229                 Tclose = 200;
230             }
231 
232             //close with timer
233             if ((count2 == Tclose) && valveOpen)
234             {
235                 closeValve();
236                 if (inaResult >= 546 && !prevFilled) prevFilled = true;
237             }
238 
239             //close valve with SG at 0.8V
240             if ((inaResult <= 546) && valveOpen)
241             {
242                 if (!prevFilled) prevFilled = true;
243                 closeValve();
244             }
245 
246             //if balloon loses pressure based off of SG re-open, from 1.1V
247             if ((inaResult > 750) && !valveOpen && prevFilled)
248             {
249                 openValve();
250             }
251 
```

```
252             //display SG value
253             if (count1 == 99)
254             {
255                 sgVolt = ((double)inaResult/1024.0)*refVol;
256
257     //          USART0.TXDATAH = (uint8_t)(sgVolt >> 8);
258     //          USART0.TXDATAL = (uint8_t)(sgVolt & 0xFF);
259
260                 dtostrf(sgVolt, 5, 2, strResult);
261                 strcat(strResult, "V ");
262
263                 for (i = 0; i < 10; i++)
264                 {
265                     if (strResult[i] != NULL)
266                     {
267                         USART0.TXDATAL = strResult[i];
268                         _delay_ms(2);
269                     }
270                 }
271             }
272         }
273     }
274
```

Appendix B

EAGLE PCB Designs

This appendix contains the schematics of the five versions of the hardware as done in EAGLE. The different versions get more involved as they go up, from version 1 just testing the H-bridge and valve, to version 5 being the complete set-up. Section B.1 covers the connections between all the elements for all the boards and section B.2 shows the layout of the final PCB.

B.1 Schematics

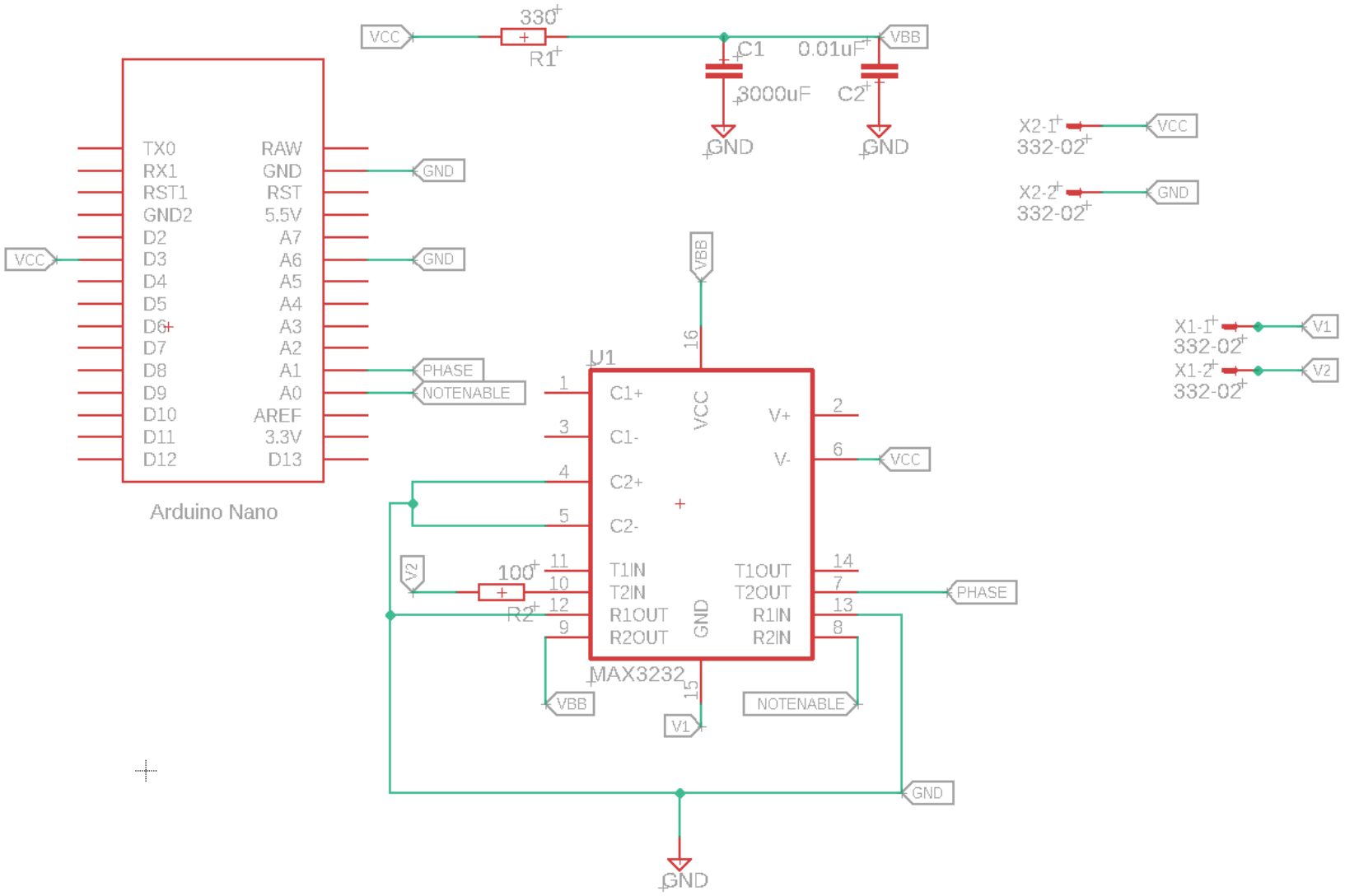


Figure B.1: PCB Version 1 Connections Schematic

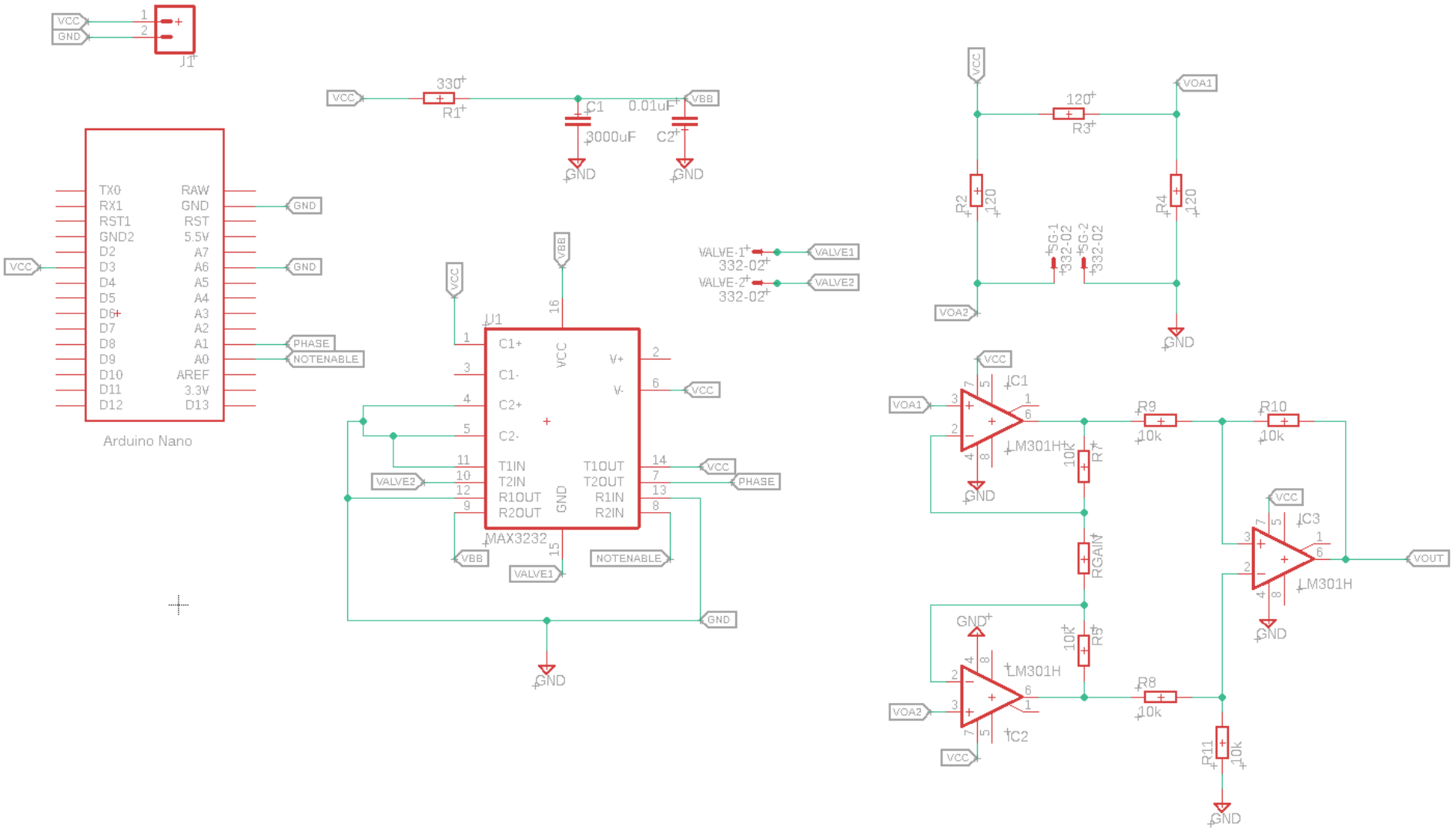


Figure B.2: PCB Version 2 Connections Schematic

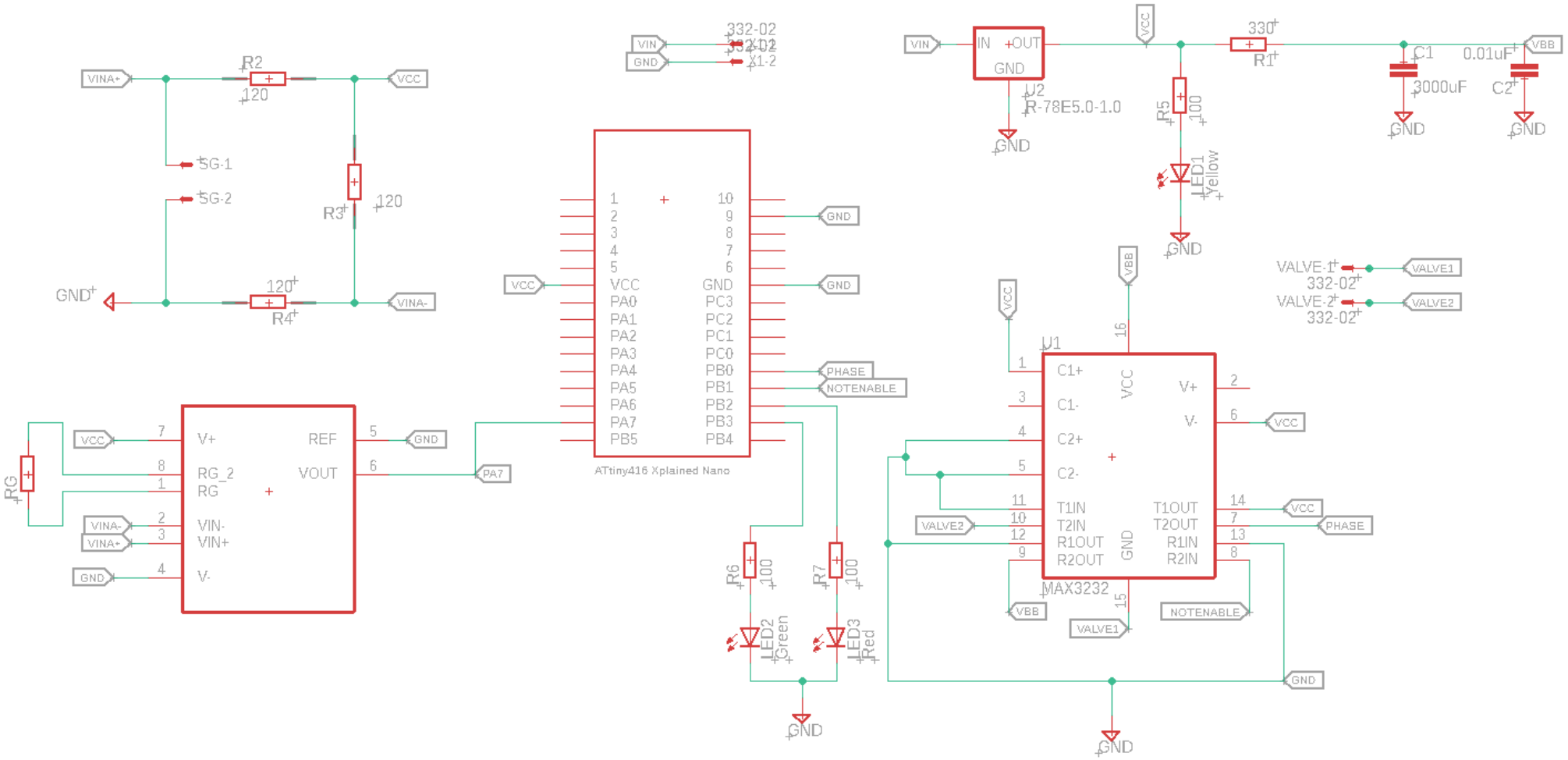


Figure B.3: PCB Version 3 Connections Schematic

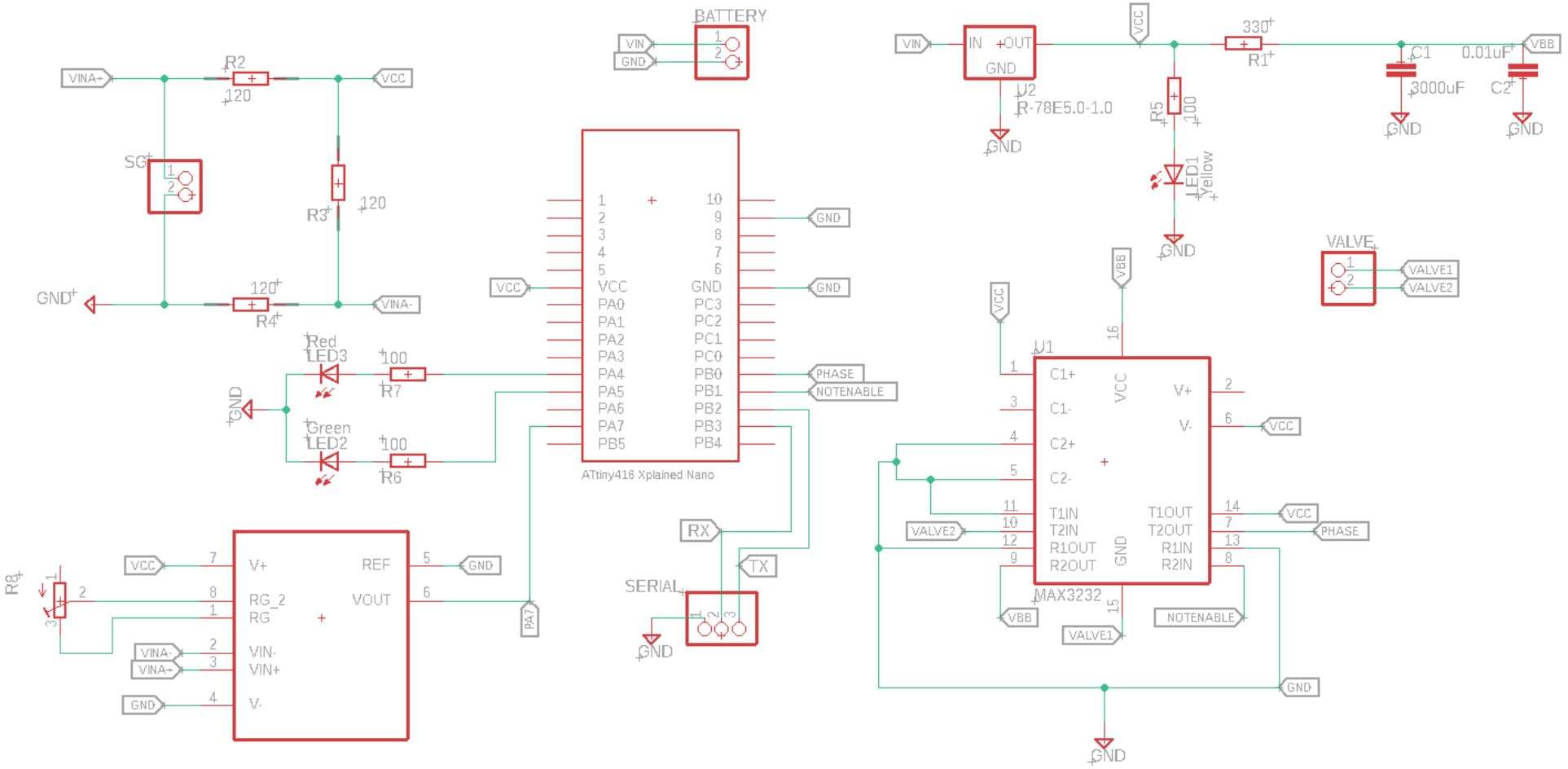


Figure B.4: PCB Version 4 Connections Schematic

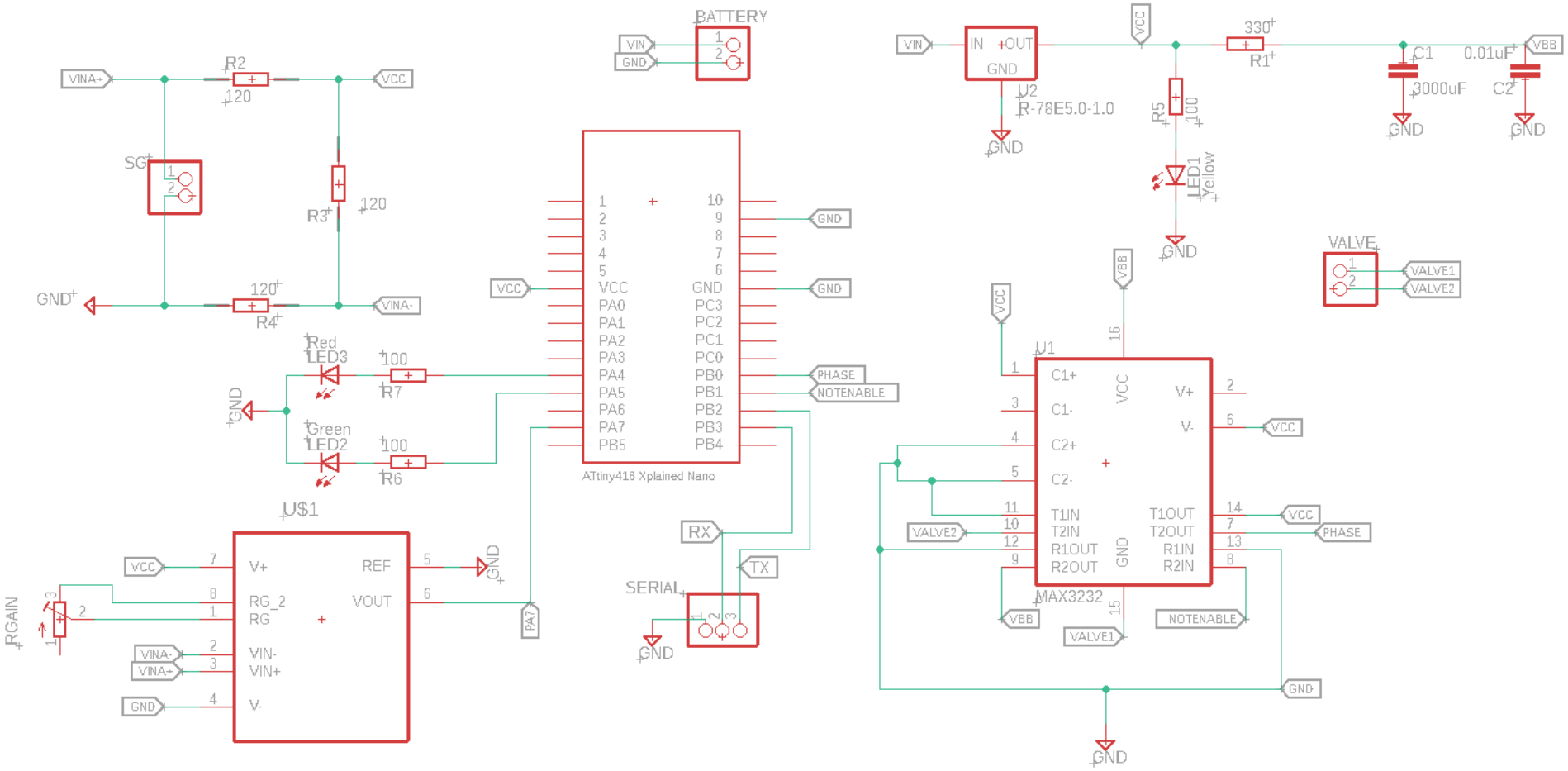


Figure B.5: PCB Version 5 Connections Schematic

B.2 Board Layouts

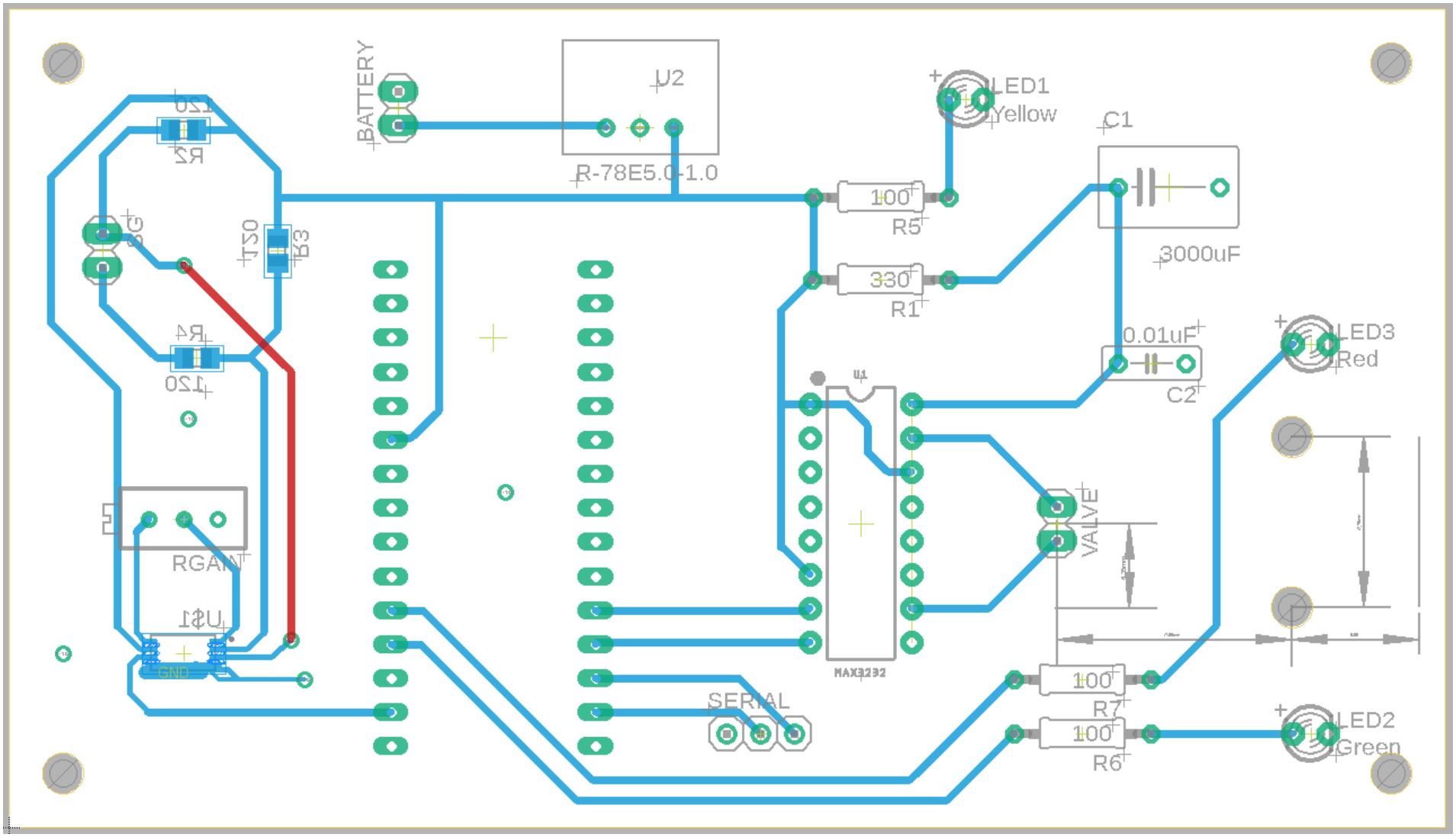


Figure B.6: PCB Version 5 Board Layout

Appendix C

MATLAB Code

C.1 Original MATLAB Sail Code

The program in this subsection is the original code for the Orbit Propagator (version 4), provided by Prof. WH Steyn to be modified for this project. The only edits made below were for formatting in this thesis.

```
% clear all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               Initialization File                                             %
%                        for                                                    %
%               Cube Sail  Orbit Propagator                                     %
%                                                                               %
%  Change Values in this file to change propagator initial condition            %
%                                                                               %
%                                                                               %
%     04/2009 Nasir Adeli modified WH Steyn 05/2009                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Constants        DO NOT CHANGE
P = 4.563E-6;          % Nominal SRP constant at 1 A.U. [N/m^2] (~1367 W/m^2)
GMe = 398600.4415e+9;           % [m^3/s^2]; JGM3
Re = 6378137;                   % Radius Earth [m]; WGS-84
Rs = 696000.0e3;                % Radius Sun [m]; Seidelmann 1992
AU = 149597870000.0;            % Astronomical unit [m]; IAU 1976
omega_Earth = 7.2921158553e-5;  % [rad/s]; Aoki 1982, NIMA 1997
store_sample_time = 10*60;      % Sample time to store sim parameters into workspace

%% Sail Parameters  ONLY MAKE CHANGES HERE
A_sail = 25;                            % (m^2)
A_drag = 5*0.03;                        % (m^2)
m_sail = 2.0;                           % (kg)
ac = inf;
I_sail = eye(3);

% Sail Optical characteristics (JPL)
r = 0.88;       s = 0.94;
Bf = 0.79;      Bb = 0.55;
ef = 0.05;      eb = 0.55;
rs = r*s;
rd = Bf*r*(1-s) + ((ef*Bf-eb*Bb)/(ef+eb))*(1-r);

%% Initialize   ONLY MAKE CHANGES HERE
Altitude = 800000;                      % Approx (m)
Rinit_ = [7182480 , 77790, 10];         % (m) @ Epoch POSAT in ECI frame
Vcirc = sqrt(GMe/(Re+Altitude));        % Approx (m/s) Velocity for a circular orbit
Vinit_ = [5.76, -1085.96, 7368.16];     % (m/s) @ Epoch POSAT in ECI frame
CTime = [2000,11,02,16,43,23];          % Sun model start @ Epoch POSAT
```

C.2 Modified Balloon Code

The following code was used to initialize the satellite and balloon parameters. This specific instance implements a 6U CubeSat, solar activity flag set to mean activity, with a sail of radii $0.5 \times 1.0 \times 1.0$ m at Sumbandila's orbit height of around 480 km.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     Initialization File                                 %
%                             for                                         %
%                  CubeSat  Orbit Propagator                              %
%                                                                         %
%          Change values in this file to                                  %
%          change propagator initial condition                            %
%                                                                         %
% 04/2009 Nasir Adeli modified WH Steyn 05/2009, AK Naude 2019            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Constants
P = 4.563E-6;          % Nominal SRP constant at 1 A.U. [N/m^2] (~1367 W/m^2)
GMe = 398600.4415e+9;          % [m^3/s^2]; JGM3
Re = 6378137;                  % Radius Earth [m]; WGS-84
Rs = 696000.0e3;               % Radius Sun [m]; Seidelmann 1992
AU = 149597870000.0;           % Astronomical unit [m]; IAU 1976
omega_Earth = 7.2921158553e-5; % [rad/s]; Aoki 1982, NIMA 1997
sample_time = 10;              % simulink ST (sec)
activity = 1;                  % 1 for mean solar activity, 0 for minimum solar
activity
omega_max = 0;                 % for maximum yaw rate (deg/sec)

% Atmospheric density values during solar mean (kg/m^3)
mean_atmos_densities = [ 1.20e0, 5.69e-07, 2.02e-09, 7.66e-10, 2.90e-10, ...
                         1.46e-10, 7.30e-11, 4.10e-11, 2.30e-11, 1.38e-11, ...
                         8.33e-12, 5.24e-12, 3.29e-12, 1.39e-12, 6.15e-13, ...
                         2.84e-13, 1.37e-13, 6.87e-14, 3.63e-14, 2.02e-14, ...
                         1.21e-14, 7.69e-15, 5.24e-15, 3.78e-15, 2.86e-15];

% Atmospheric density values during solar minimum (kg/m^3)
min_atmos_densities = [ 1.20e0, 5.71e-07, 1.90e-09, 6.42e-10, 2.18e-10, ...
                        9.64e-11, 4.27e-11, 2.14e-11, 1.07e-11, 5.83e-12, ...
                        3.17e-12, 1.81e-12, 1.04e-12, 3.68e-13, 1.40e-13, ...
                        5.76e-14, 2.61e-14, 1.32e-14, 7.55e-15, 4.81e-15, ...
                        3.34e-15, 2.47e-15, 1.90e-15, 1.50e-15, 1.20e-15];

% Atmospheric density values during solar maximum (kg/m^3)
max_atmos_densities = [ 1.20e0, 5.67e-07, 2.21e-09, 9.21e-10, 3.84e-10, ...
                        2.12e-10, 1.17e-10, 7.17e-11, 4.39e-11, 2.85e-11, ...
                        1.85e-11, 1.25e-11, 8.43e-12, 4.05e-12, 2.03e-12, ...
                        1.05e-12, 5.63e-13, 3.08e-13, 1.73e-13, 9.95e-14, ...
                        5.88e-14, 3.57e-14, 2.25e-14, 1.46e-14, 9.91e-15];

% Altitudes from SMAD (km)
altitudes_SMAD = [  1, 100, 150, 175, 200, ...
                  225, 250, 275, 300, 325, ...
                  350, 375, 400, 450, 500, ...
                  550, 600, 650, 700, 750, ...
                  800, 850, 900, 950, 1000];

altitudes_finer = 5:5:1000;

% Using SMAD altitude values with interpolation for finer resolution
interpolated_mean_densities = interp1(altitudes_SMAD, mean_atmos_densities,
altitudes_finer, 'pchip');
interpolated_max_densities = interp1(altitudes_SMAD, max_atmos_densities,
altitudes_finer, 'pchip');
interpolated_min_densities = interp1(altitudes_SMAD, min_atmos_densities,
altitudes_finer, 'pchip');

%% Balloon Parameters
a = 0.5;                    % (m)
```

```
60    b = 1.0;                    % (m)
61    c = 1.0;                    % (m)
62    m_balloon = 0.5;            % (kg)
63    m_sat = 8.0;                % (kg) - 4 for 3U, 8 for 6U, 16 for 12U
64    m_tot = m_balloon + m_sat;
65    ac = inf;
66    I_sail = eye(3);
67
68    % Balloon Optical characteristics (JPL)
69    r = 0.88;       s = 0.94;
70    Bf = 0.79;      Bb = 0.55;
71    ef = 0.05;      eb = 0.55;
72    rs = r*s;
73    rd = Bf*r*(1-s) + ((ef*Bf-eb*Bb)/(ef+eb))*(1-r);
74
75    %% Initialize
76    % Sumbandila is nr 1, Sunsat is nr 2, 950km is nr 3
77    Vinit_ = [-1.319335e+03, 3.741929e+03, -6.505140e+03]; % (m/s) nr 1 in ECI
78    Rinit = [-3.457551e+06, 4.819777e+06, 3.462959e+06];   % (m) nr 1 in ECI
79    Rnorm = norm(Rinit);
80    Vcirc = norm(Vinit_);
81    Rmag = GMe/(Vcirc*Vcirc);
82    Altitude = Rmag - Re;
83    Rinit_ = Rmag*Rinit/Rnorm;
84    CTime = [2019,08,29,07,24,40];          % Sun model start "Epoch"
85    h0 = Altitude/1000;
```

C.3 Modified Sail Code

This code is the modified code for this project's simulation as in section C.2; however, it implements a sail as a deorbit device instead of a balloon. The orbit height and CubeSat parameters are the same as those in the code for a balloon, with sail length and width of 2 m each.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Initialization File                                          %
%                          for                                                %
%             CubeSat Sail Orbit Propagator                                   %
%                                                                             %
%      Change Values in this file to change propagator initial condition      %
%                                                                             %
%                                                                             %
%     04/2009 Nasir Adeli modified WH Steyn 05/2009, AK NaudÃ© 2019           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Constants
P = 4.563E-6;                   % Nominal SRP constant at 1 A.U. [N/m^2] (~1367 W/m^2)
GMe = 398600.4415e+9;           % [m^3/s^2]; JGM3
Re = 6378137;                   % Radius Earth [m]; WGS-84
Rs = 696000.0e3;                % Radius Sun [m]; Seidelmann 1992
AU = 149597870000.0;            % Astronomical unit [m]; IAU 1976
omega_Earth = 7.2921158553e-5;  % [rad/s]; Aoki 1982, NIMA 1997
store_sample_time = 10*60;      % Sample time to store sim parameters into workspace
sample_time = 10;               % simulink ST (sec)
activity = 1;                   % 1 for mean solar activity, 0 for minimum solar
activity

% Atmospheric density values during solar mean (kg/m^3)
mean_atmos_densities = [  1.20e0, 5.69e-07, 2.02e-09, 7.66e-10, 2.90e-10, ...
                          1.46e-10, 7.30e-11, 4.10e-11, 2.30e-11, 1.38e-11, ...
                          8.33e-12, 5.24e-12, 3.29e-12, 1.39e-12, 6.15e-13, ...
                          2.84e-13, 1.37e-13, 6.87e-14, 3.63e-14, 2.02e-14, ...
                          1.21e-14, 7.69e-15, 5.24e-15, 3.78e-15, 2.86e-15];

% Atmospheric density values during solar minimum (kg/m^3)
min_atmos_densities = [  1.20e0, 5.71e-07, 1.90e-09, 6.42e-10, 2.18e-10, ...
                         9.64e-11, 4.27e-11, 2.14e-11, 1.07e-11, 5.83e-12, ...
                         3.17e-12, 1.81e-12, 1.04e-12, 3.68e-13, 1.40e-13, ...
                         5.76e-14, 2.61e-14, 1.32e-14, 7.55e-15, 4.81e-15, ...
                         3.34e-15, 2.47e-15, 1.90e-15, 1.50e-15, 1.20e-15];

% Atmospheric density values during solar maximum (kg/m^3)
max_atmos_densities = [  1.20e0, 5.67e-07, 2.21e-09, 9.21e-10, 3.84e-10, ...
                         2.12e-10, 1.17e-10, 7.17e-11, 4.39e-11, 2.85e-11, ...
                         1.85e-11, 1.25e-11, 8.43e-12, 4.05e-12, 2.03e-12, ...
                         1.05e-12, 5.63e-13, 3.08e-13, 1.73e-13, 9.95e-14, ...
                         5.88e-14, 3.57e-14, 2.25e-14, 1.46e-14, 9.91e-15];

% Altitudes from SMAD (km)
altitudes_SMAD = [  1, 100, 150, 175, 200, ...
                  225, 250, 275, 300, 325, ...
                  350, 375, 400, 450, 500, ...
                  550, 600, 650, 700, 750, ...
                  800, 850, 900, 950, 1000];

altitudes_finer = 5:5:1000;

% Using SMAD altitude and density values with interpolation for finer resolution
interpolated_mean_densities = interp1(altitudes_SMAD, mean_atmos_densities,
altitudes_finer, 'pchip');
interpolated_max_densities = interp1(altitudes_SMAD, max_atmos_densities,
altitudes_finer, 'pchip');
interpolated_min_densities = interp1(altitudes_SMAD, min_atmos_densities,
altitudes_finer, 'pchip');

%% Sail Parameters
```

```
58      t_sail = 0.001;                         % (m)
59      l_sail = 2.0;                           % (m)
60      h_sail = 2.0;                           % (m)
61      A_sail = l_sail * h_sail;               % (m^2)
62      m_sail = 0.5;                           % (kg)
63      m_sat = 4.0;                            % (kg)
64      m_tot = m_sail + m_sat;
65      ac = inf;
66      I_sail = eye(3);
67
68      % Sail Optical characteristics (JPL)
69      r = 0.88;       s = 0.94;
70      Bf = 0.79;      Bb = 0.55;
71      ef = 0.05;      eb = 0.55;
72      rs = r*s;
73      rd = Bf*r*(1-s) + ((ef*Bf-eb*Bb)/(ef+eb))*(1-r);
74
75      %% Initialize
76      % Sumbandila is nr 1, Sunsat is nr 2, 950km is nr 3
77      Vinit_ = [-1.319335e+03, 3.741929e+03, -6.505140e+03];  % (m/s) @ Epoch nr 1 in ECI
        frame
78      Rinit = [-3.457551e+06, 4.819777e+06, 3.462959e+06];    % (m) @ Epoch nr 1 in ECI frame
79      Rnorm = norm(Rinit);
80      Vcirc = norm(Vinit_);
81      Rmag = GMe/(Vcirc*Vcirc);
82      Altitude = Rmag - Re;
83      Rinit_ = Rmag*Rinit/Rnorm;
84      CTime = [2019,08,29,07,24,40];                          % Sun model start "Epoch"
85      h0 = Altitude/1000;
```

C.4 Orbit Position and Velocity Code

This code is used to retrieve the satellite's orbit position and velocity in ECI from a TLE file. The comments at the end of the code are the orbit parameters used in this project.

```
% SGP4 orbit verification
% 26/07/2019 - WH Steyn

clear all
R2D = 180.0/pi;
IDtle = 25636; % input('TLE ID of satellite = ');

Offset = input('Offset (sec) from TLE epoch = ');
[P,A] = orbit3(0,IDtle);        % Initialize SGP4
[P,A] = orbit3(Offset,IDtle);   % Get initial IRF position and velocity vector
XYZpos = num2str([P(4),P(5),P(6)]*1e3,'%10.6e\n')
XYZvel = num2str([P(7),P(8),P(9)]*1e3,'%10.6e\n')


% Saved Values at Epoch +1000 sec
% Sumbandila
% ID = 35870
%     Vinit_ = [-1.319335e+03, 3.741929e+03, -6.505140e+03];  % (m/s) @ Epoch nr 1 in ECI frame
%     Rinit = [-3.457551e+06, 4.819777e+06, 3.462959e+06];    % (m) @ Epoch nr 1 in ECI frame
%
%
% SUNSAT
% ID = 25636
%     Vinit_ = [-9.524421e+02, -2.288863e+03, -7.093367e+03]; % (m/s) @ Epoch no. 2 in ECI frame
%     Rinit = [4.641462e+06, 4.845821e+06, -2.294144e+06];    % (m) @ Epoch no. 2 in ECI frame
% Date = 1999-04-30T09:16:46.458 + 20 yrs + 4 mnths
%
% 950 km
%     Vinit_ = [-5.118960e+03, 3.728266e+03, 3.788593e+03];   % (m/s) @ Epoch no. 3 in ECI frame
%     Rinit = [2.397433e+06, -2.920019e+06, 6.147029e+06];    % (m) @ Epoch no. 3 in ECI frame
% Date = 1999-04-30T09:16:46.458 + 1000 sec
%
%
% ISS (only used for Epoch)
% ID = 25544
% Vinit = [-4.804273e+03, -3.603983e+03, -4.753304e+03]
% Rinit = [-5.294472e+06, 2.750065e+06, 3.252766e+06]
% TLE Epoch = 8/29/2019 7:08:00 AM UTC + 1000 sec
```

Appendix D

Simulink Blocks

D.1 Original Simulink Sail Blocks

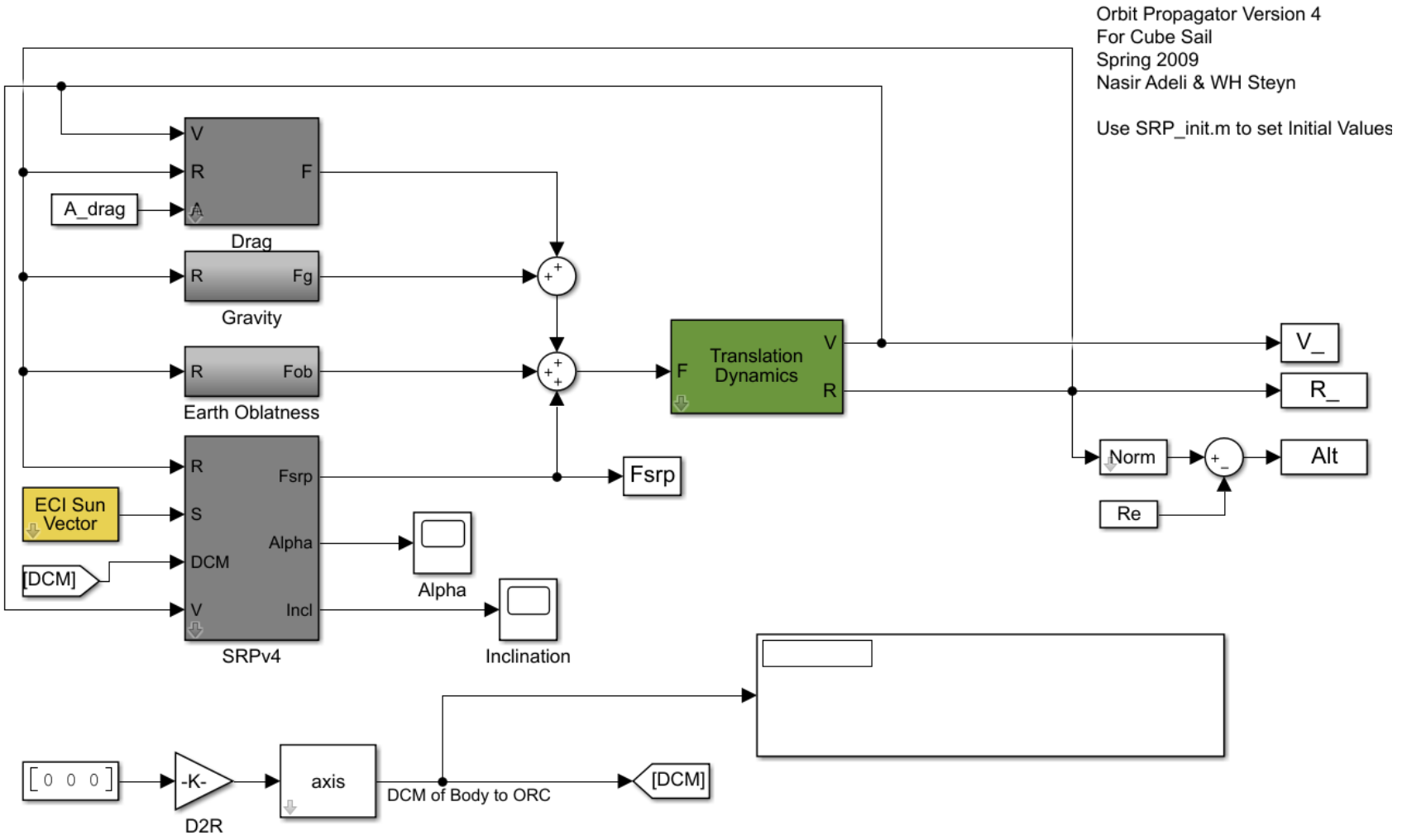


Figure D.1: The original code's Simulink blocks.

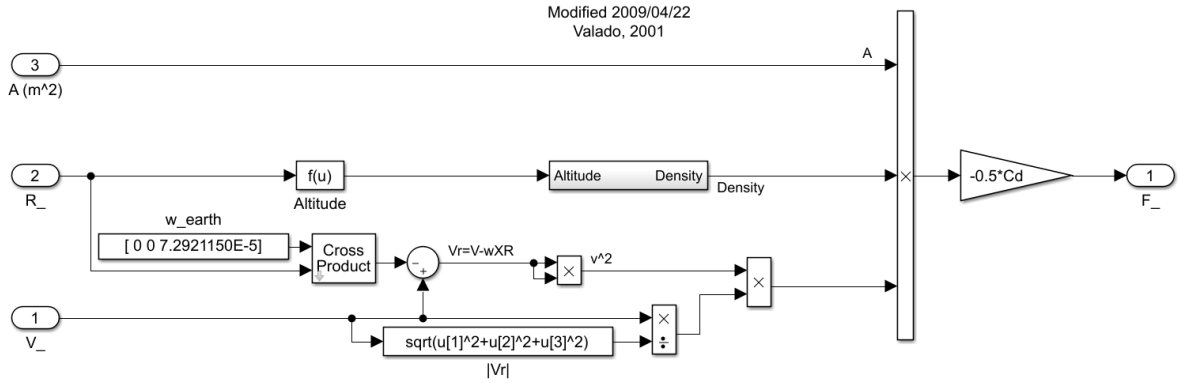


Figure D.2: Inside the original code's "Drag" block.

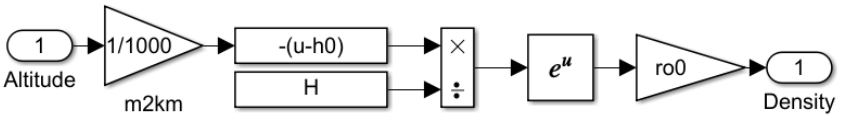


Figure D.3: Inside the Altitude to Density block in the original code's "Drag" block.

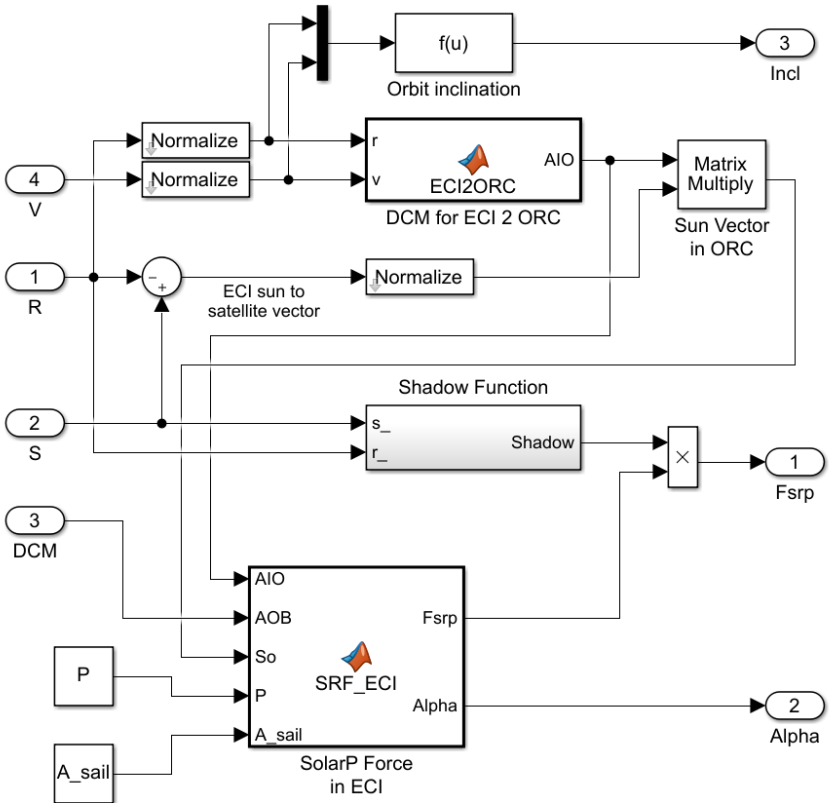


Figure D.4: Inside the original code's "SRPv4" block.

This code is for the original code's "SolarP Force in ECI" block inside the "SRPv4" block.

```
function [Fsrp, Alpha] = SRF_ECI(AIO,AOB,So,P,A_sail)
% This block computes the Solar radiation pressure force in ECI
% Alpha is sun angle to normal of sail surface

Sb = AOB*So;                % Sun vector in SBC
Cos_alpha = abs(Sb(2));     % Sail normal parallel to body Y-axis
Sin_alpha = sqrt(1-Sb(2)*Sb(2));
Fn = 1.83*P*A_sail*Cos_alpha*Cos_alpha;
Ft = 0.17*P*A_sail*Cos_alpha*Sin_alpha;
sn = sqrt(Sb(1)*Sb(1)+Sb(3)*Sb(3));
if sn < 1e-6
    sn = 1e-6;
end
if Sb(2) < 0                % Impact sail front
    Fs = [Ft*Sb(1)/sn; -Fn; Ft*Sb(3)/sn];
    Alpha = acos(Cos_alpha);
else                        % Impact sail back
    Fs = [Ft*Sb(1)/sn; Fn; Ft*Sb(3)/sn];
    Alpha = -acos(Cos_alpha);
end
Fsrp = AIO'*AOB'*Fs;        % Transform SRP force from SRB to ORC to ECI
Alpha = Alpha*180/pi;       % Convert Alpha angle to deg
```

D.2 Modified Balloon Blocks

D.2.1 Static Solar Activity

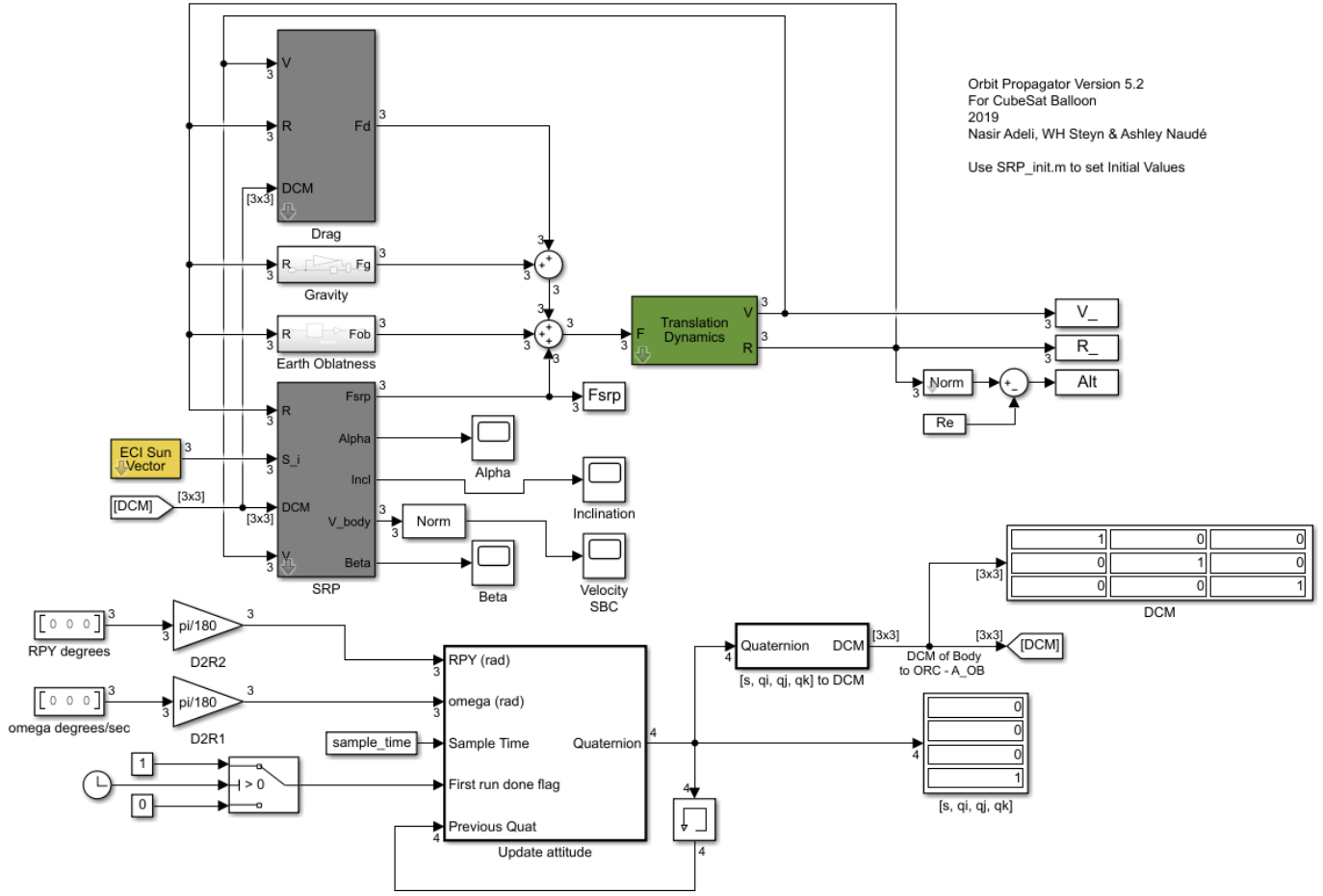


Figure D.5: All the Simulink blocks used for the static solar activity cases.

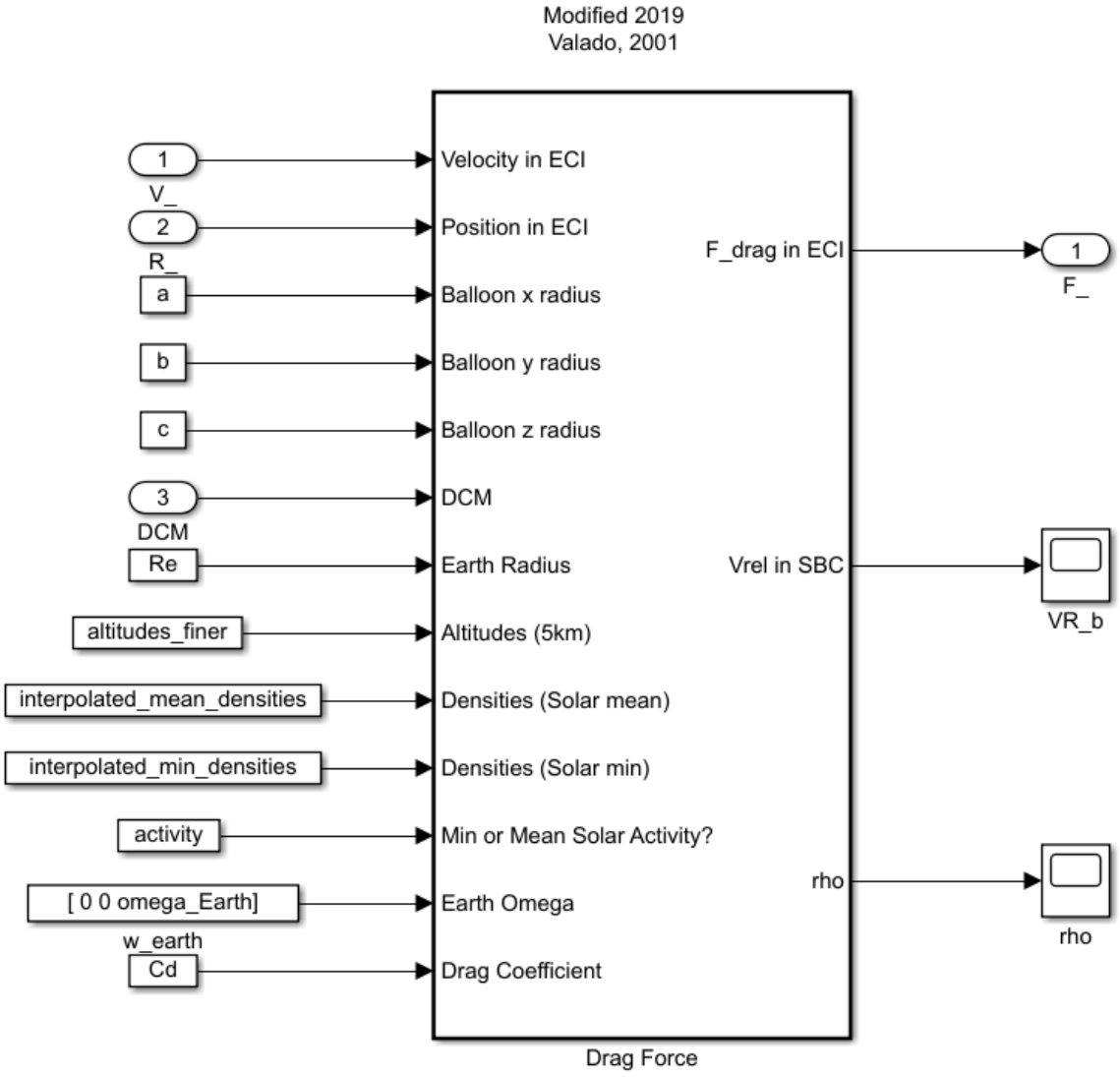


Figure D.6: The blocks inside the static case's "Drag" block.

The following code is for the static case's "Drag Force" block inside the "Drag" block.

```
function [F_drag, Vr_b, rho] = fcn(Vi, Ri, a, b, c, DCM, Re, altitudes_finer,
interpolated_mean_densities, interpolated_min_densities, activity, We, Cd)
coder.extrinsic('set_param');

% Find drag area ------------------------------------------------------
Ax = b*c*pi;
Ay = a*c*pi;
Az = a*b*pi;

A_drag = [Ax; Ay; Az];

% Find atmospheric density --------------------------------------------
% R_mag is current orbit altitude (Re + Rsat) (km)
% R_sat is the current orbit height (km)
Rmag = norm(Ri)/1000;
Rsat = Rmag - (Re/1000);

% Check for mean solar activity or minimum solar activity
if activity == 1
    densities = interpolated_mean_densities;
else
    densities = interpolated_min_densities;
end

Rsat_rounded = round(Rsat/5)*5;

if Rsat_rounded <= 1000
    check = find(altitudes_finer == Rsat_rounded, 1);
    if isempty(check)
        n = 1;
    else
        n = check(1);
    end
else
    n = length(altitudes_finer);
end

if Rsat < 150
    set_param('SRP_balloon', 'SimulationCommand', 'stop');
end

rho = densities(n);

% Find velocity -------------------------------------------------------
% Will have to transform velocity vector to body coordinates
R_normalised = Ri/norm(Ri);
V_normalised = Vi/norm(Vi);

AIO = zeros(3,3);               %DCM for ECI to ORC
AIO(3,:) = -R_normalised';
vXr = cross(V_normalised, R_normalised);
AIO(2,:) = vXr';
AIO(1,:) = cross(R_normalised, vXr)';

% Must use atmospheric velocity (V_relative)
wXr = cross(We, Ri);
Vr_i = Vi - wXr;

% Transform to Body Coords
Vr_b = DCM * AIO * Vr_i;

% Calculate drag force ------------------------------------------------
Vsquared = Vr_b .* Vr_b;
F_body = -0.5*Cd*rho * (A_drag.*Vsquared) .* sign(Vr_b);
% Transform back to Inertial Coords
F_drag = AIO'*DCM'*F_body;
end
```

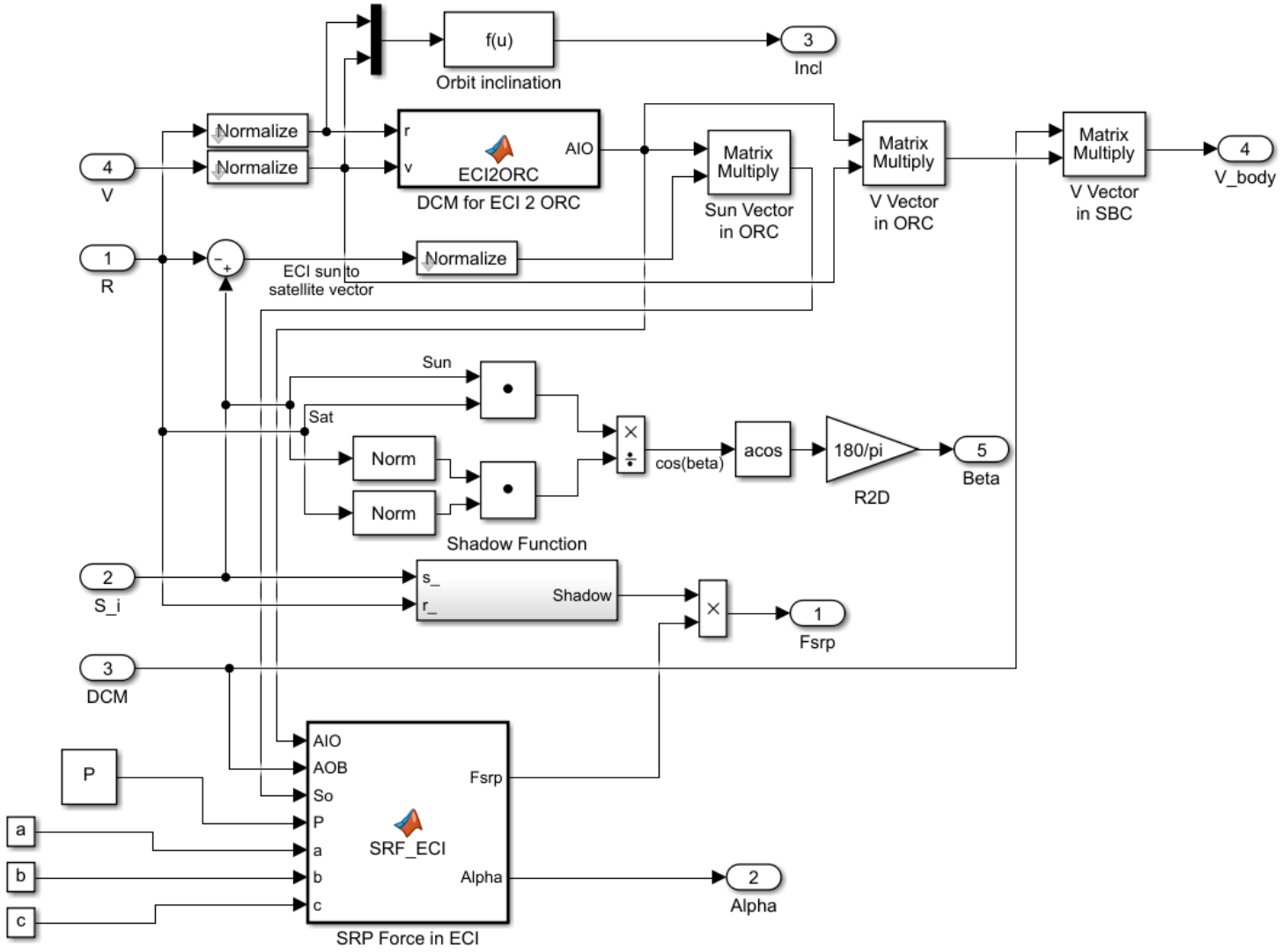


Figure D.7: The blocks inside the "SRP" block in the static case.

The code below is for the static "SRP Force in ECI" block inside the "SRP" block. This code is the same for static solar activity cases as well as the dynamic solar activity case.

```
function [Fsrp, Alpha] = SRF_ECI(AIO,AOB,So,P,a, b, c)
% This block computes the Solar radiation pressure force in ECI
% Alpha is sun angle to normal of sail surface

%AIO - Inertial (ECI) to Orbit (ORC)
%AOB (DCM) - Orbit (ORC) to Body (SBC)

Sb = AOB*So;                % Sun vector in SBC from ORC

Ax = b*c*pi;
Ay = a*c*pi;
Az = a*b*pi;

A_solar = Ax*Sb(1) + Ay*Sb(2) + Az*Sb(3);

Cos_alpha = abs(Sb(1));                     % Balloon normal parallel to body x-axis
Sin_alpha = sqrt(1-Sb(1)*Sb(1));
Fn = 1.83*P*A_solar*Cos_alpha*Cos_alpha;    % Normal force component
Ft = 0.17*P*A_solar*Cos_alpha*Sin_alpha;    % Tangential force compontent
sn = sqrt(Sb(2)*Sb(2)+Sb(3)*Sb(3));         % Sun norm

% Prevent division by zero
if sn < 1e-6
    sn = 1e-6;
end

Fs = [Fn; Ft*Sb(2)/sn; Ft*Sb(3)/sn];
Alpha = acos(Cos_alpha);

Fsrp = AIO'*AOB'*Fs;        % Transform SRP force from SRB to ORC to ECI
Alpha = Alpha*180/pi;       % Convert Alpha angle to deg
```

This code is for the static case's "Update Attitude" block.

```
function q = fcn(RPY_rad, omega_rad, sample_time, first_run_done, qk_old)
coder.extrinsic('angle2quat');

roll = RPY_rad(1);
pitch = RPY_rad(2);
yaw = RPY_rad(3);

wx = omega_rad(1);
wy = omega_rad(2);
wz = omega_rad(3);

q = zeros(4,1);
cos_matrix = zeros(4,4);
identity4 = diag([1.0 1.0 1.0 1.0]);

if wx == 0 && wy == 0 && wz == 0
    % All omega values are zero, no change in RPY
    rotate_flag = 0;
else
    % At least one body rate change
    rotate_flag = 1;
end

if first_run_done == 0 || rotate_flag == 0
    %RPY to Quat, rotation order 213 or YXZ
    qktemp = angle2quat(roll, pitch, yaw, 'YXZ');           % Aerospace standard, 1x4
matrix
    qk = [qktemp(2); qktemp(3); qktemp(4); qktemp(1)];      % Restructure for
satellite standard, 4x1 matrix
else
```

```
29          qk = qk_old;
30          %disp(qk_old)
31      end
32
33      qk_norm = norm(qk);
34
35      % Fixed RPY
36      if rotate_flag == 0
37          qkplus1 = qk;
38      % RPY changing
39      else
40          if first_run_done == 0
41              qkplus1 = qk;
42          else
43              omega_norm = norm(omega_rad);
44
45              %disp(omega_norm)
46
47              % Using the definition from Steyn's ADCS notes (Satelltite Systems)
48              omega_matrix = [  0,  wz, -wy, wx;
49              -wz,   0,  wx, wy;
50              wy, -wx,   0, wz;
51              -wx, -wy, -wz, 0];
52
53              %disp(omega_matrix)
54
55              cos_matrix = cos(omega_norm*sample_time/2.0)*identity4 ;
        %4x4 matrix
56              sin_matrix = (1.0/omega_norm)*sin(omega_norm*sample_time/2.0)*omega_matrix;
        %4x4 matrix
57
58              %disp(cos_matrix)
59              %disp(sin_matrix)
60
61              qkplus1 = (cos_matrix + sin_matrix)*(qk/qk_norm);       %From Steyn
62          end
63      end
64
65      q = qkplus1;
66      %disp(q)
67      end
```

D.2.2 Dynamic Solar Activity

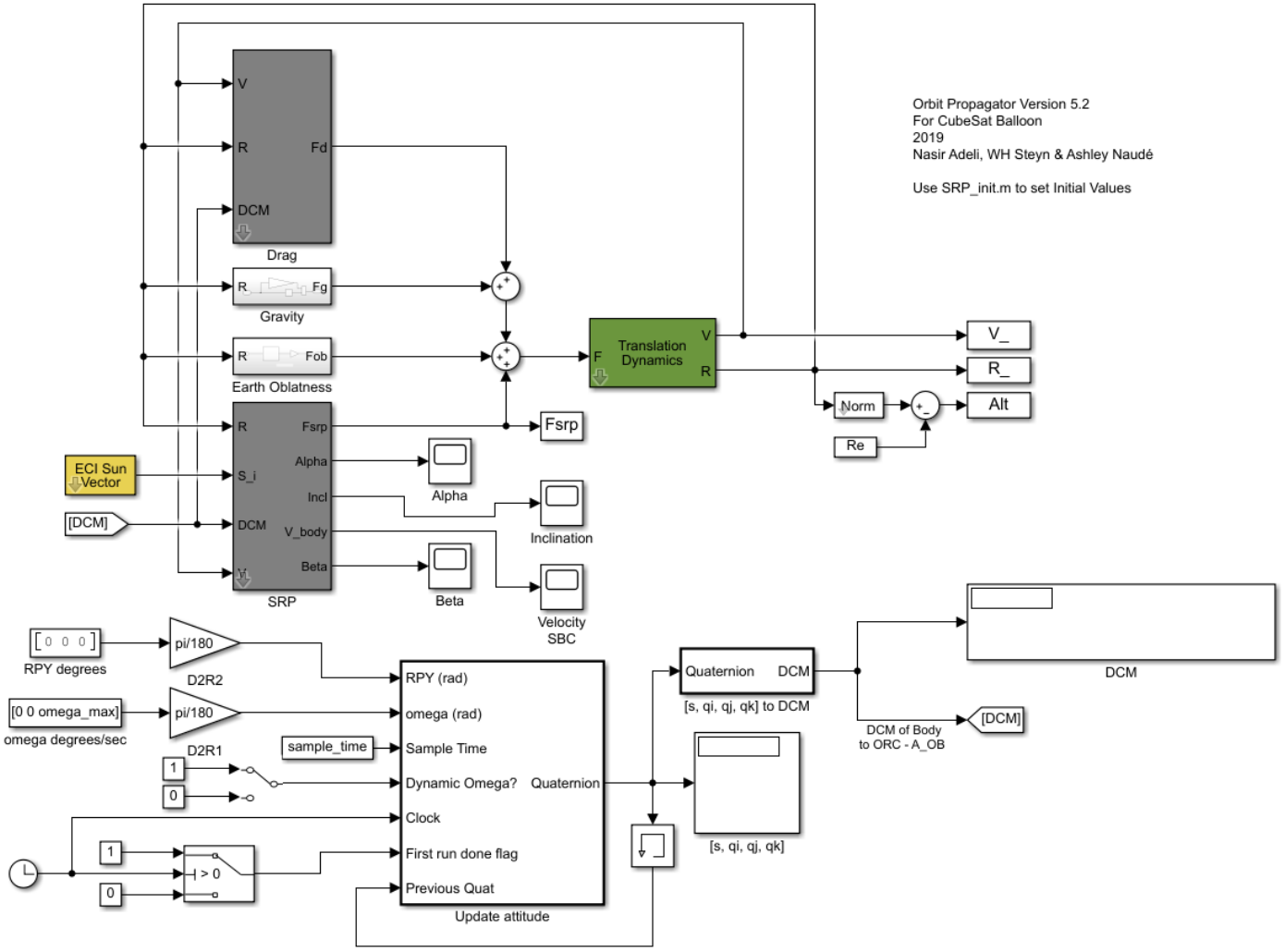


Figure D.8: All the Simulink blocks used for the dynamic solar activity case.

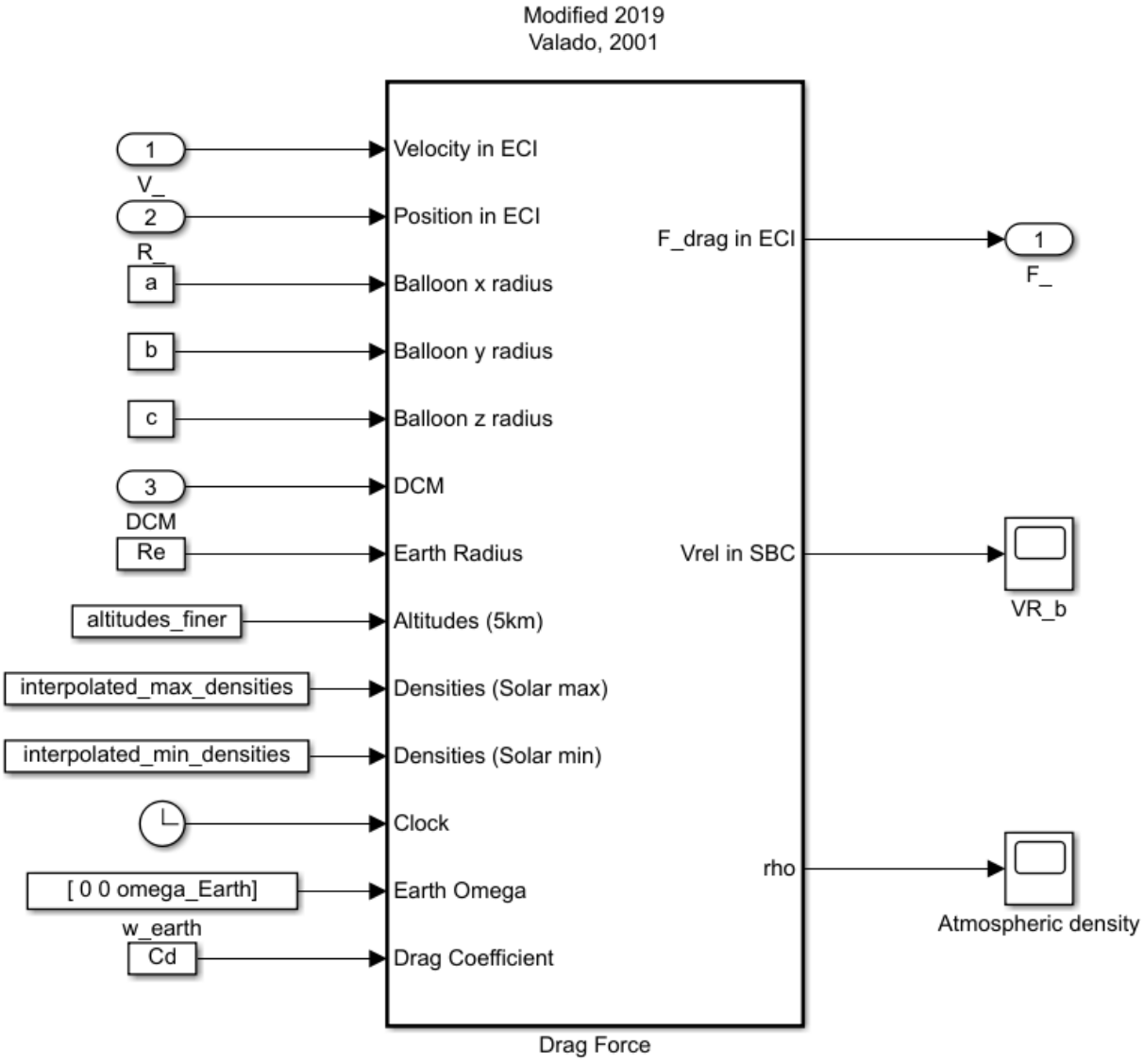


Figure D.9: The blocks inside the dynamic case's "Drag" block.

The following code is for the dynamic case's "Drag Force" block inside the "Drag" block.

```
function [F_drag, Vr_b, rho] = fcn(Vi, Ri, a, b, c, DCM, Re, altitudes_finer,
interpolated_max_densities, interpolated_min_densities, time, We, Cd)
coder.extrinsic('set_param');

% Find drag area ------------------------------------------------------------
Ax = b*c*pi;
Ay = a*c*pi;
Az = a*b*pi;

A_drag = [Ax; Ay; Az];

% Find atmospheric density --------------------------------------------------
% R_mag is current orbit altitude (Re + Rsat) (km)
% R_sat is the current orbit height (km)
Rmag = norm(Ri)/1000;
Rsat = Rmag - (Re/1000);


Rsat_rounded = round(Rsat/5)*5;

if Rsat_rounded <= 1000
    check = find(altitudes_finer == Rsat_rounded, 1);
    if isempty(check)
        n = 1;
    else
        n = check(1);
    end
else
    n = length(altitudes_finer);
end

if Rsat < 150
    set_param('SRP_balloon', 'SimulationCommand', 'stop');
end

amplitude = (interpolated_max_densities(n) - interpolated_min_densities(n))/2;
offset = interpolated_min_densities(n) + amplitude;

rho = -amplitude * cos(2*pi*time/(11*365*24*60*60)) + offset;

% Find velocity -------------------------------------------------------------
% Will have to transform velocity vector to body coordinates
R_normalised = Ri/norm(Ri);
V_normalised = Vi/norm(Vi);

AIO = zeros(3,3);               %DCM for ECI to ORC
AIO(3,:) = -R_normalised';
vXr = cross(V_normalised, R_normalised);
AIO(2,:) = vXr';
AIO(1,:) = cross(R_normalised, vXr)';

% Must use atmospheric velocity (V_relative)
wXr = cross(We, Ri);
Vr_i = Vi - wXr;

% Transform to Body Coords
Vr_b = DCM * AIO * Vr_i;


% Calculate drag force ------------------------------------------------------
Vsquared = Vr_b .* Vr_b;
F_body = -0.5*Cd*rho * (A_drag.*Vsquared) .* sign(Vr_b);
% Transform back to Inertial Coords
F_drag = AIO'*DCM'*F_body;
end
```

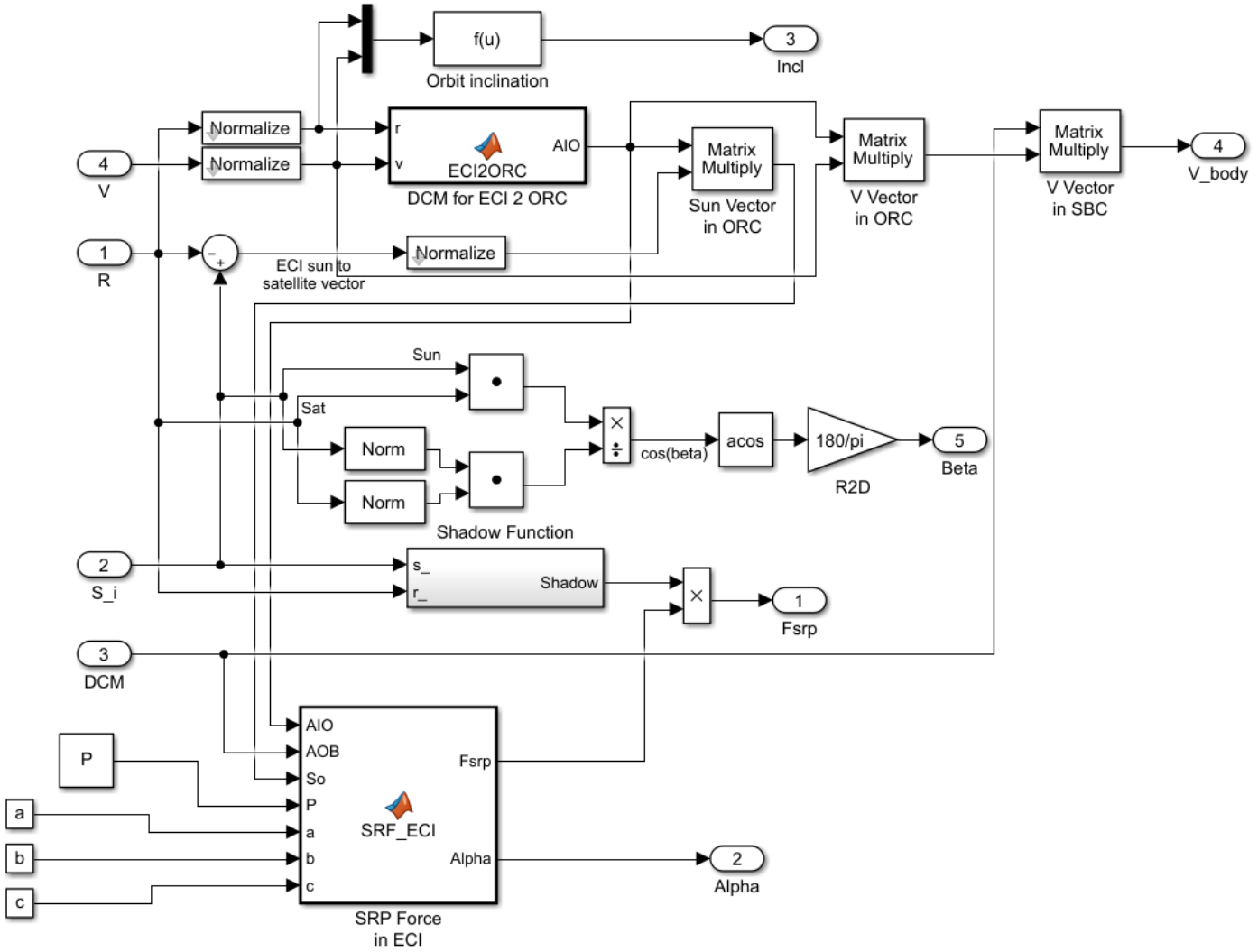


Figure D.10: The blocks inside the "SRP" block in the dynamic case.

For consistency, this is the code in the dynamic case's "SRP Force in ECI" block inside the "SRP" block. As mentioned previously, it is the same in both static and dynamic cases.

```
function [Fsrp, Alpha] = SRF_ECI(AIO,AOB,So,P,a, b, c)
% This block computes the Solar radiation pressure force in ECI
% Alpha is sun angle to normal of sail surface

%AIO - Inertial (ECI) to Orbit (ORC)
%AOB (DCM) - Orbit (ORC) to Body (SBC)

Sb = AOB*So;                % Sun vector in SBC from ORC

Ax = b*c*pi;
Ay = a*c*pi;
Az = a*b*pi;

A_solar = Ax*Sb(1) + Ay*Sb(2) + Az*Sb(3);

Cos_alpha = abs(Sb(1));                     % Balloon normal parallel to body x-axis
Sin_alpha = sqrt(1-Sb(1)*Sb(1));
Fn = 1.83*P*A_solar*Cos_alpha*Cos_alpha;    % Normal force component
Ft = 0.17*P*A_solar*Cos_alpha*Sin_alpha;    % Tangential force component
sn = sqrt(Sb(2)*Sb(2)+Sb(3)*Sb(3));         % Sun norm

% Prevent division by zero
if sn < 1e-6
    sn = 1e-6;
end

Fs = [Fn; Ft*Sb(2)/sn; Ft*Sb(3)/sn];
Alpha = acos(Cos_alpha);

Fsrp = AIO'*AOB'*Fs;        % Transform SRP force from SRB to ORC to ECI
Alpha = Alpha*180/pi;       % Convert Alpha angle to deg
```

The following code is the dynamic "Update Attitude" block's code. It is only slightly different from the static case, in that it accounts for dynamic yawing motions.

```
function q = fcn(RPY_rad, omega_rad, sample_time, dynamic, clock, first_run_done,
qk_old)
coder.extrinsic('angle2quat');

roll = RPY_rad(1);
pitch = RPY_rad(2);
yaw = RPY_rad(3);

wx = omega_rad(1);
wy = omega_rad(2);
wz = omega_rad(3);

q = zeros(4,1);
cos_matrix = zeros(4,4);
identity4 = diag([1.0 1.0 1.0 1.0]);

if wx == 0 && wy == 0 && wz == 0
    % All omega values are zero, no change in RPY
    rotate_flag = 0;
else
    % At least one body rate change
    rotate_flag = 1;
end

if first_run_done == 0 || rotate_flag == 0
    %RPY to Quat, rotation order 213 or YXZ
    qktemp = angle2quat(roll, pitch, yaw, 'YXZ');          % Aerospace standard, 1x4
matrix
```

```
27          qk = [qktemp(2); qktemp(3); qktemp(4); qktemp(1)];      % Restructure for
    satellite standard, 4x1 matrix
28      else
29          qk = qk_old;
30          %disp(qk_old)
31      end
32
33      qk_norm = norm(qk);
34
35      % Fixed RPY
36      if rotate_flag == 0
37          qkplus1 = qk;
38
39      % RPY changing
40      else
41          % Dynamic yaw
42          if dynamic == 1
43              arg = pi*clock/4 + pi/4;
44              wz = wz * sin(arg);
45          end
46
47          if first_run_done == 0
48              qkplus1 = qk;
49          else
50              omega_norm = norm(omega_rad);
51
52              %disp(omega_norm)
53
54              % Using the definition from Steyn's ADCS notes (Satelltite Systems)
55              omega_matrix = [  0,  wz, -wy, wx;
56                              -wz,   0,  wx, wy;
57                               wy, -wx,   0, wz;
58                              -wx, -wy, -wz, 0];
59
60              %disp(omega_matrix)
61
62              cos_matrix = cos(omega_norm*sample_time/2.0)*identity4 ;
    % 4x4 matrix
63              sin_matrix = (1.0/omega_norm)*sin(omega_norm*sample_time/2.0)*omega_matrix;
    % 4x4 matrix
64
65              %disp(cos_matrix)
66              %disp(sin_matrix)
67
68              qkplus1 = (cos_matrix + sin_matrix)*(qk/qk_norm);      % From Steyn
69          end
70      end
71
72      q = qkplus1;
73      %disp(q)
74      end
```

D.3 Modified Sail Blocks

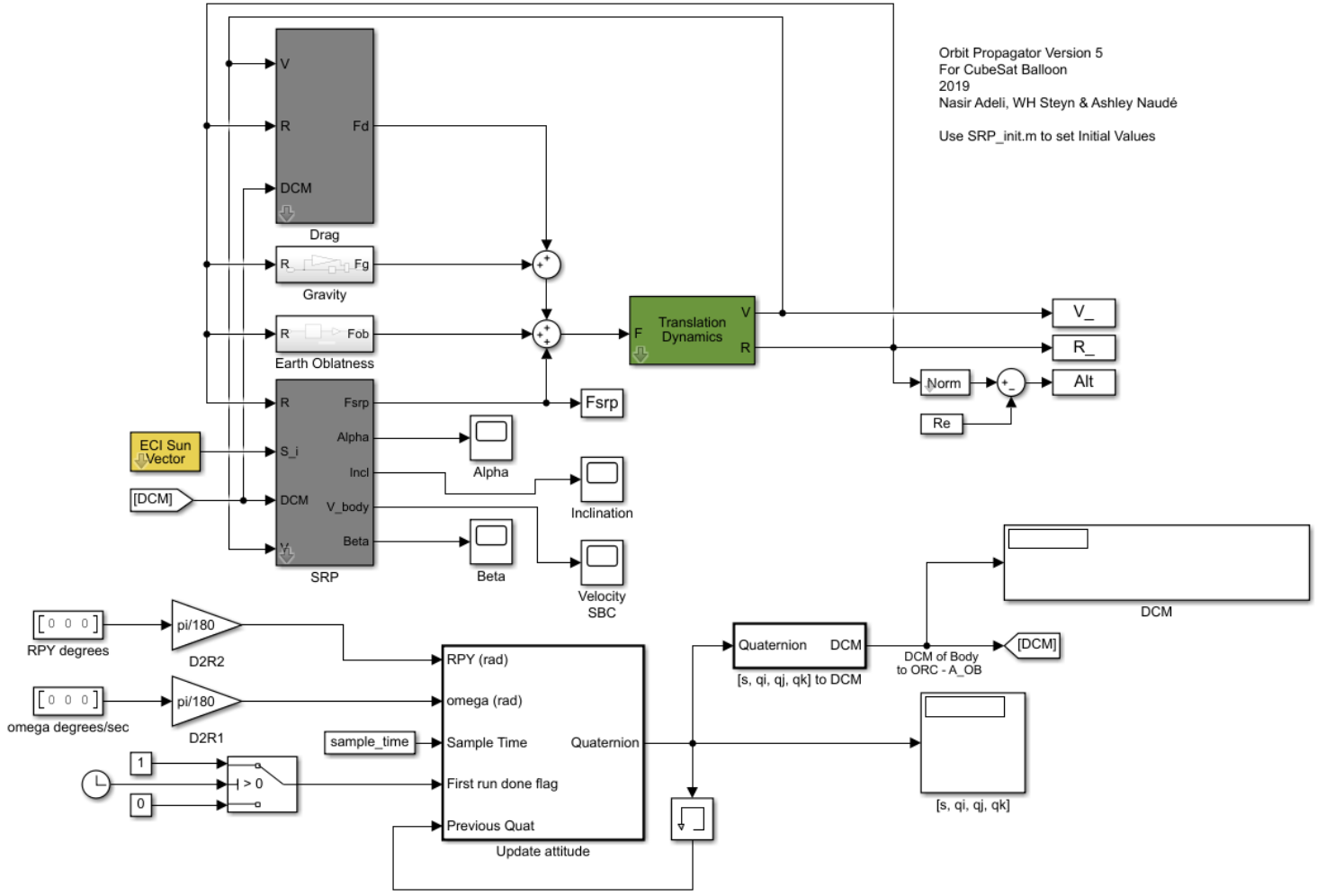


Figure D.11: All the Simulink blocks used for the sail device cases.

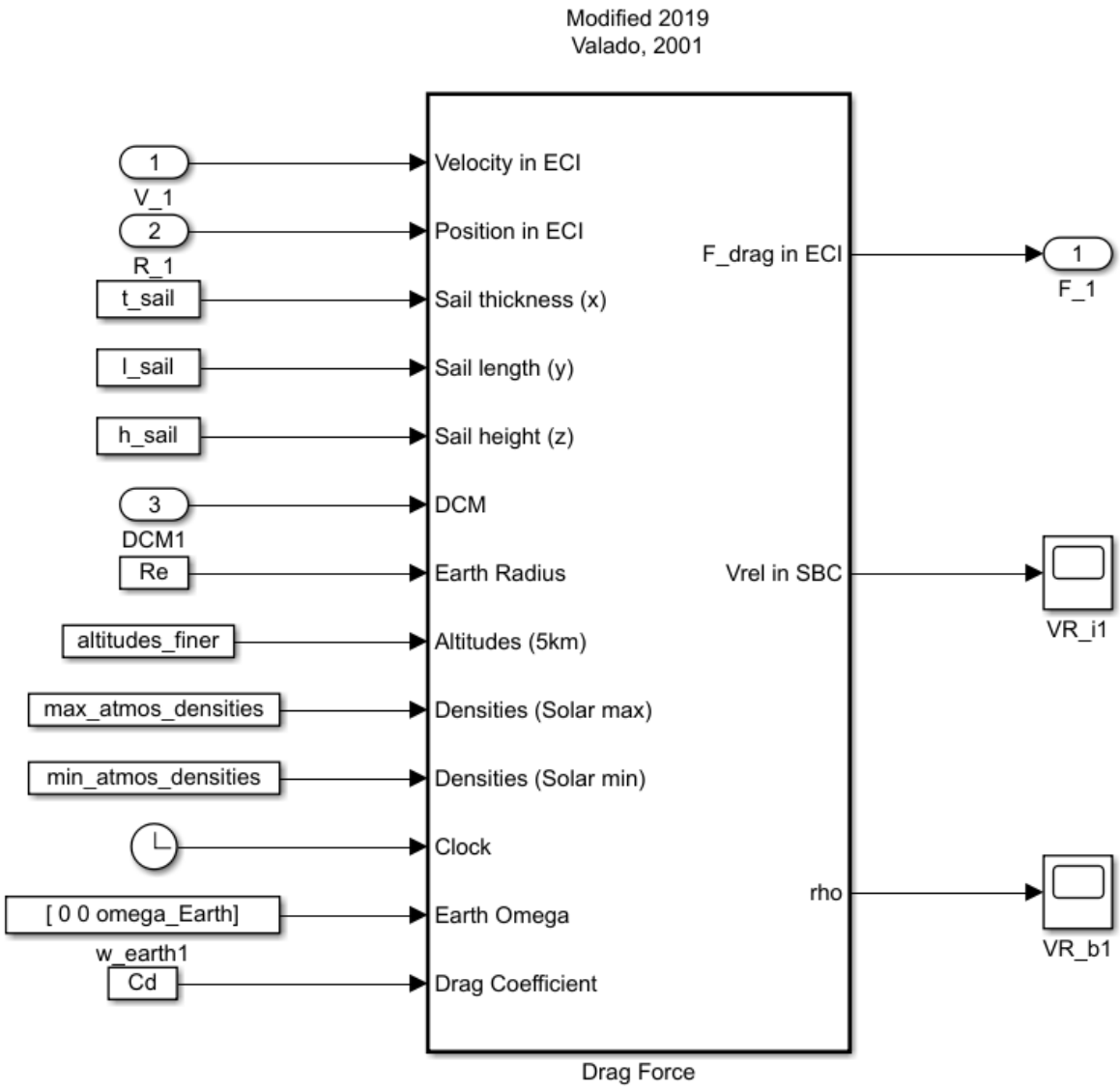


Figure D.12: The blocks inside the sail case's "Drag" block.

The following code is for the sail case's "Drag Force" block inside the "Drag" block. Due to the sail working with dynamic atmospheric conditions, the code is almost the same as that from the dynamic balloon's case.

```
function [F_drag, Vr_b, rho] = fcn(Vi, Ri, thickness, length, height, DCM, Re,
altitudes_finer, interpolated_max_densities, interpolated_min_densities, time, We, Cd)
coder.extrinsic('set_param');

% Find drag area ------------------------------------------------------------
Ax = length*height;
Ay = thickness*height;
Az = thickness*length;

A_drag = [Ax; Ay; Az];

% Find atmospheric density --------------------------------------------------
% R_mag is current orbit altitude (Re + Rsat) (km)
% R_sat is the current orbit height (km)
Rmag = norm(Ri)/1000;
Rsat = Rmag - (Re/1000);

Rsat_rounded = round(Rsat/5)*5;

if Rsat_rounded <= 1000
    check = find(altitudes_finer == Rsat_rounded, 1);
    if isempty(check)
        n = 1;
    else
        n = check(1);
    end
else
    n = length(altitudes_finer);
end

if Rsat < 150
    set_param('SRP_balloon', 'SimulationCommand', 'stop');
end

amplitude = (interpolated_max_densities(n) - interpolated_min_densities(n))/2;
offset = interpolated_min_densities(n) + amplitude;

rho = -amplitude * cos(2*pi*time/(11*365*24*60*60)) + offset;

% Find velocity -------------------------------------------------------------
% Will have to transform velocity vector to body coordinates
R_normalised = Ri/norm(Ri);
V_normalised = Vi/norm(Vi);

AIO = zeros(3,3);               %DCM for ECI to ORC
AIO(3,:) = -R_normalised';
vXr = cross(V_normalised, R_normalised);
AIO(2,:) = vXr';
AIO(1,:) = cross(R_normalised, vXr)';

% Must use atmospheric velocity (V_relative)
wXr = cross(We, Ri);
Vr_i = Vi - wXr;

% Transform to Body Coords
Vr_b = DCM * AIO * Vr_i;

% Calculate drag force ------------------------------------------------------
Vsquared = Vr_b .* Vr_b;
F_body = -0.5*Cd*rho * (A_drag.*Vsquared) .* sign(Vr_b);
% Transform back to Inertial Coords
F_drag = AIO'*DCM'*F_body;
end
```

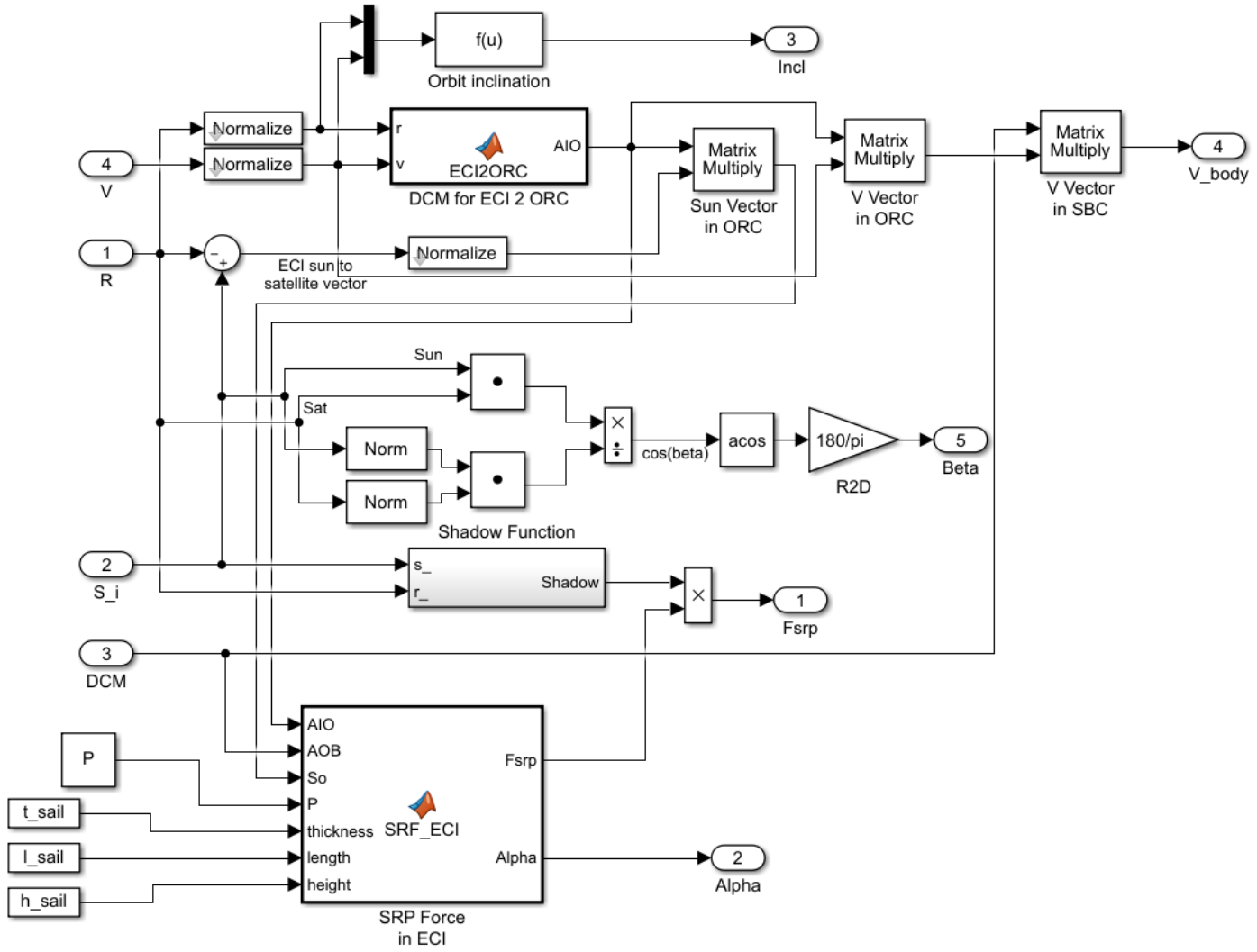


Figure D.13: The blocks inside the "SRP" block in the sail case.

The code below is for the sail "SRP Force in ECI" block inside the "SRP" block. This code is very similar to the other version, just with a different area calculation.

```
function [Fsrp, Alpha] = SRF_ECI(AIO,AOB,So,P,thickness, length, height)
% This block computes the Solar radiation pressure force in ECI
% Alpha is sun angle to normal of sail surface

%AIO - Inertial (ECI) to Orbit (ORC)
%AOB (DCM) - Orbit (ORC) to Body (SBC)

Sb = AOB*So;                % Sun vector in SBC from ORC

Ax = length*height;
Ay = thickness*height;
Az = thickness*length;

A_solar = Ax*Sb(1) + Ay*Sb(2) + Az*Sb(3);

Cos_alpha = abs(Sb(1));                     % Sail normal parallel to body x-axis
Sin_alpha = sqrt(1-Sb(1)*Sb(1));
Fn = 1.83*P*A_solar*Cos_alpha*Cos_alpha;    % Normal force component
Ft = 0.17*P*A_solar*Cos_alpha*Sin_alpha;    % Tangential force compontent
sn = sqrt(Sb(2)*Sb(2)+Sb(3)*Sb(3));         % Sun norm

% Prevent division by zero
if sn < 1e-6
    sn = 1e-6;
end

Fs = [Fn; Ft*Sb(2)/sn; Ft*Sb(3)/sn];
Alpha = acos(Cos_alpha);

Fsrp = AIO'*AOB'*Fs;        % Transform SRP force from SRB to ORC to ECI
Alpha = Alpha*180/pi;       % Convert Alpha angle to deg
```

This code is for the sail case's "Update Attitude" block. It is the same as the one from the static balloon's case.

```
function q = fcn(RPY_rad, omega_rad, sample_time, first_run_done, qk_old)
coder.extrinsic('angle2quat');

roll = RPY_rad(1);
pitch = RPY_rad(2);
yaw = RPY_rad(3);

wx = omega_rad(1);
wy = omega_rad(2);
wz = omega_rad(3);

q = zeros(4,1);
cos_matrix = zeros(4,4);
identity4 = diag([1.0 1.0 1.0 1.0]);

if wx == 0 && wy == 0 && wz == 0
    % All omega values are zero, no change in RPY
    rotate_flag = 0;
else
    % At least one body rate change
    rotate_flag = 1;
end

if first_run_done == 0 || rotate_flag == 0
    %RPY to Quat, rotation order 213 or YXZ
    qktemp = angle2quat(roll, pitch, yaw, 'YXZ');           % Aerospace standard, 1x4
matrix
    qk = [qktemp(2); qktemp(3); qktemp(4); qktemp(1)];      % Restructure for
satellite standard, 4x1 matrix
else
```

```
29          qk = qk_old;
30          %disp(qk_old)
31      end
32
33      qk_norm = norm(qk);
34
35      % Fixed RPY
36      if rotate_flag == 0
37          qkplus1 = qk;
38      % RPY changing
39      else
40          if first_run_done == 0
41              qkplus1 = qk;
42          else
43              omega_norm = norm(omega_rad);
44
45              %disp(omega_norm)
46
47              % Using the definition from Steyn's ADCS notes (Satelltite Systems)
48              omega_matrix = [  0,  wz, -wy, wx;
49                           -wz,   0,  wx, wy;
50                            wy, -wx,   0, wz;
51                           -wx, -wy, -wz, 0];
52
53              %disp(omega_matrix)
54
55              cos_matrix = cos(omega_norm*sample_time/2.0)*identity4 ;
        %4x4 matrix
56              sin_matrix = (1.0/omega_norm)*sin(omega_norm*sample_time/2.0)*omega_matrix;
        %4x4 matrix
57
58              %disp(cos_matrix)
59              %disp(sin_matrix)
60
61              qkplus1 = (cos_matrix + sin_matrix)*(qk/qk_norm);       %From Steyn
62          end
63      end
64
65      q = qkplus1;
66      %disp(q)
67      end
```